Von der Carl-Friedrich-Gauß-Fakultät für Mathematik und Informatik
der Technischen Universität Braunschweig
genehmigte Dissertation zur Erlangung
des Grades eines Doktor-Ingenieurs (Dr.-Ing.)

CREATION, MANAGEMENT AND PUBLICATION OF DIGITAL DOCUMENTS USING STANDARD COMPONENTS ON THE INTERNET

Marco Zens

Datum der mündlichen Prüfung: 17. November 2006

1. Referent	Prof. Dr. Dieter W. Fellner
2. Referent	Prof. Dr. Hermann Maurer
eingereicht am:	16. August 2004

Abstract

Today, digital documents play a steadily increasing role within our society. The Internet as well as the World Wide Web, offering the opportunity for easy and efficient distributed collaboration, additionally speed up this process. It is therefore as much promising as necessary to develop, to implement and to evaluate new systems to support workflows based on digital documents and Web technology. However, it is important not only to replace traditional (e.g. paper-based) workflows with their electronic counterpart, but also to improve them by exploiting the new possibilities.

In this thesis, following an introduction to the field of digital documents and digital libraries, three consecutive application scenarios about the successful application of digital workflows will be presented. First, the research project *GoldenGate* focuses on the creation of and the access to structured documents. Then, the project *EG Online* is discussed as an example scenario for information and distributed collaboration with the help of Web Services. Finally, the results from the research project *Managing Conference Proceedings* are presented, which implements a complete electronic creation, managing and publication workflow based on digital documents, using ideas and techniques developed during the previous two projects.

All these projects have led to new systems and procedures which were and still are used successfully in different environments. For example *EG Online* is the central tool in the daily work of the *Eurographics Association* and several large scientific conferences as well as workshops are supported by the *MCP* system every year. The presentation and the discussion of the achieved results and of the experiences gained form a major part of this work.

Zusammenfassung

Heutzutage spielen digitale Dokumente in unserer Gesellschaft eine ständig zunehmende Rolle. Das Internet und das World Wide Web beschleunigen diesen Prozess noch zusätzlich, in dem sie die Gelegenheit für eine einfache und doch effiziente Zusammenarbeit von verteilten Standorten aus bieten. Es ist daher ebenso viel versprechend wie auch notwendig, neue Systeme zu entwickeln, zu implementieren und zu evaluieren, welche auf digitalen Dokumenten und auf Web-Technologie basierende Workflow-Prozesse unterstützen können. Es ist jedoch wichtig, traditionelle (also z.B. auf Papier basierende) Arbeitsprozesse nicht einfach nur durch ihre elektronischen Gegenstücke zu ersetzen, sondern sie auch durch die Ausnutzung der neuen Möglichkeiten zu verbessern.

Im Rahmen dieser Dissertation werde ich, nach einer Einführung in das Gebiet der digitalen Dokumente und der digitalen Bibliotheken, drei aufeinander aufbauende Anwendungsszenarien zum erfolgreichen Einsatz digitaler Workflows vorstellen. Zunächst konzentriert sich das Forschungsprojekt *GoldenGate* auf die Erstellung und auf den Zugriff auf strukturierte Dokumente. Im Anschluss wird das *EG Online*-Projekt diskutiert, das als beispielhaftes Szenario zur Verbreitung von Informationen und zur verteilten Zusammenarbeit, beides unter Verwendung von Web Services, dient. Schließlich werden die Resultate des Forschungsprojekts *Managing Conference Proceedings* präsentiert, in dessen Verlauf ein vollständig elektronischer Erzeugungs-, Verwaltungs- und Publikationsprozess implementiert worden ist. Dabei werden vor allem Techniken und Ideen verwendet, die in den zwei vorhergehenden Projekten entwickelt wurden.

Alle diese Projekte haben zu neuen Systemen und Prozeduren geführt, welche in unterschiedlichen Umgebungen erfolgreich eingesetzt wurden und auch noch eingesetzt werden. So ist z.B. *EG Online* das zentrale Werkzeug in der täglichen Arbeit der *Eurographics Association* und mehrere große, wissenschaftliche Konferenzen sowie Workshops werden jedes Jahr durch das *MCP*-System unterstützt. Die Darstellung sowie die Diskussion der erzielten Ergebnisse und der gewonnenen Erfahrungen bilden einen großen Anteil dieser Arbeit.

Acknowledgments

I would like to take the opportunity to thank everyone who supported me during the last years and contributed to the research projects discussed or to this thesis itself, be it with helpful comments, with long hours of programming, with encouragement or patience.

At first, I have to mention my supervisor Dieter Fellner who gave me the chance to work on such interesting projects. Without his ideas, his knowledge, his comments and his continuing support the results presented here would not have been accomplished.

Additionally, I want to explicitly thank all current and former members of the Digital Library Lab and the Computer Graphics Group in Bonn and later in Braunschweig, especially René Berndt (for contributing essential parts to all three projects), Carsten Oberscheid (for his outstanding work during *GoldenGate*) and Thomas Günther (for proof-reading this thesis, together with René).

Also I acknowledge the funding and the cooperation of the *Deutsche Forschungsgemeinschaft* for *GoldenGate*, the support of the *BMBF* for *Managing Conference Proceedings* and the helpful comments of numerous members of the *Eurographics Association* who have to "live" with the *EG Online* system which we implemented.

Last but not least, a sincere "Thank you!" goes to my parents, my friends and especially to Heike for her patience and for her subtle ways to drive me on. *;-)*

Contents

Chapter 1

Introduction

1.1 Motivation

Today, digital documents play a major role in our society, with their importance steadily increasing. With the World Wide Web spreading like wildfire, with the wide usage of email or electronic messaging services and with networked computers being placed in almost every office or private home, digital documents have become a central issue for academia, industry and society at large. But one will be confronted with uncertainness as soon as one asks for the definition of an "electronic document" or if one only wants to know what kind of requirements it has to fulfill in order to be superior to a printed document. It all starts with trivial questions like "What is a document?" or "In what format should it be stored?". The answers to the these questions immediately lead to more issues which need to be discussed.

When preparing a concept for a system which is to use digital documents, one soon recognizes that many questions arise from the simple idea of storing a specific information in a document. Important problems are, for example, the *structure* of the content, the *syntax* to represent it and to what file *format* this can be mapped to – it is only much later that one asks for the optical (or acoustical etc.) appearance of the document, the so-called *presentation*. This rather quickly leads to sophisticated techniques for the storage of documents and even more important techniques for the retrieval of and the access to these documents. A quite crucial question is also how such a document is created in the first place and how it is published, if required.

The field of research named *Digital Libraries*, as a part of Computer Science (but also Library Science, Media Science and so on) deals with these and other important issues. It is important to stress that Digital Library (or *DL*) is neither only a synonym for the electronic version of a bookshelf nor an institution owning many digital books (which is one aspect, of course). Instead, the term more generally refers to the creation, the managing and the distribution of digital documents in any form. This young (at least under that name) field of research is very wide and comprises many traditional and highly relevant disciplines, like Databases or Information Retrieval to name a few.

1.2 Thesis Contribution

This dissertation wants to give an overview about which possibilities digital documents can offer today and how this can be achieved. The central issue is, what requirements have to be fulfilled by the documents themselves on one hand and by the deployed tools on the other hand in order to meet the demand for a digital advantage. The final goal is not only to replace the usual "analog" workflow processes ("analog" as opposite of "digital", here meaning for example paper-based) but also to improve them by the application of electronic media.

This thesis concentrates on the creation, the management, and the publication of digital documents in both, intranet and Internet environments. Additionally, it will be shown how the desired improvements

can be easily and efficiently achieved with the help of Internet technologies combined with the usage of standard tools. The research projects *GoldenGate*, *EG Online*, *Managing Conference Proceedings* and the resulting applications which will be dealt with in the later chapters convincingly show that this approach is very promising. Especially, the basic idea of building complex systems and applications by combining existing standard software components has proven to be superior to alternate approaches – like building monolithic systems from scratch. In that way, only the "glue" between the many building blocks of the complete architecture has to be newly implemented (b.t.w. this has been postulated already for a long time as a modern programming paradigm under the name of *component-based software engineering*). Another successful example is the approach to use Web Services for access and cooperation in a distributed environment.

The applications resulting from the research projects mentioned above have been constantly improved and extended over several years. Today, some of them have turned into successful services delivering impressive results. In the cooperation with users, we often entered new areas of research and application. With this dissertation I want to pass on the rich set of experiences gained over the last years, as these can serve as a valuable insight for future developments. Because of this, the discussion of the mentioned projects will take up a major part of the coming sections.

1.3 Thesis Outline

This thesis is continued in Chapter 2 "Digital Documents" with an overview about the current situation. The most important terms and notions will be defined and many important file formats for the different types of electronic media will be discussed. Later sections introduce the reader to the many possibilities of creating, storing and accessing digital documents, which prepares for the more detailed description of the research projects following.

Chapter 3 introduces the research project "GoldenGate" which is the first of three application scenarios about a modern and successful usage of digital document workflows. The focus here lies on the creation of and the access to structured electronic documents. Also, a new method to build complex systems by the combination of standard office components with Internet tools is developed.

The "EG Online" system is the second application scenario presented in Chapter 4. *EG Online* is a collection of Web Services used by the *Eurographics Association* for distributed digital information, collaboration and administration.

Chapter 5 discusses the research project "Managing Conference Proceedings" as a third application scenario. It can be seen as a capstone to the previous chapters, as here the component architecture of *GoldenGate* and the Web Services of *EG Online* are combined. The *MCP* system supports the submission, the reviewing and the publication of papers at scientific events. It is regularly being used and serves as a successful example for a complete electronic creation, managing and publication workflow based on digital documents.

Chapter 6 gives a summary of this thesis. The major achievements of the research projects discussed will be summarized, followed by a look towards future work which can, has to and most likely will be done.

Chapter 2

Digital Documents

This chapter intends to give the "big picture" on digital documents. Beginning with the most important file formats (including the new star on the horizon: *XML*, see Section 2.1.5), I will show how documents can be created either by authors or automatically (Section 2.2), how they can be stored and, even more important, how to access them (including searching and retrieving of documents, see Section 2.3). This chapter thus lays the foundation for the discussion of several applications in the forthcoming sections.

2.1 Document Formats

When one starts thinking about digital documents, what first comes to mind are text documents, e.g. created with *Microsoft Word* or maybe prepared in Adobe's *Portable Document Format* (PDF). But text is only one part of the digital document-world, though an important one. Also other media like audio and video are to be considered as documents. These thoughts lead to an even wider notion of *Generalized (Digital) Documents* or even to a necessary definition of the term "document" itself (see in Section 2.1.1 below).

Besides the media type (i.e. "What will the document contain?"), there are other important questions which have to be answered before deciding upon which document format to choose for a specific application. For example, whether the document will only be stored or processed further, whether it must be human readable or it will only be interpreted by a machine etc. It is necessary to decide, whether the document format should represent the structure or the presentation (i.e. the image, the layout) of the contained information (see Section 2.1.2) and if additional data (so called *Metadata*, see in 2.1.6) should be kept. Other issues are proprietary vs. open formats and how far in the future the document must be readable.

These questions will be shortly discussed in the following sections, together with possible solutions or at least with pointers to more in-depth articles. It is important to state that generally the best format doesn't exist. Instead, it is our task to find the best suited one for a document in a given environment, at a specific time.

2.1.1 Generalized Documents

In this section some definitions for terms and ideas which will be used very often throughout this thesis will be given. Some of these are widely accepted while others are still being discussed and might change from time to time or from discipline to discipline.

Let's start simple: what is a *document*? One of the many online dictionaries[1] in the Internet gives the following answer:

[1] http://www.infoplease.com/dictionary.html

1. a written or printed paper furnishing information or evidence, as a passport, deed, bill of sale, or bill of lading; a legal or official paper.
2. any written item, as a book, article, or letter, especially of a factual or informative nature.
3. a computer data file.
4. *Archaic.* evidence; proof.

Many people would have answered something along the lines of No. 2, because the notion of 'document' is often identified with 'text document'. Interesting is the term *informative* which gives a hint to another, less strict definition. The same applies for statement No. 3. We will come back to both issues soon. Another, often heard definition is based on what one can do with a document:

> A *document* is a unit which is collected and stored in a library.

This is already a somewhat broader view, because a library (at least today) stores other media besides written or printed paper, like audio and video materials. What will be used here is a definition which is independent from the media-type, the purpose of a document or the cultural background, but based on the content. When using this reasoning, often the term *generalized document* is used.

Definition 1 *A (generalized)* document *is a unit of information.*

When defined in this way, a *digital document* (or "computer data file") simply becomes a special case of a generalized document. Following [FE00]:

Definition 2 *A* digital document *is a self-contained unit of information. Its content is digitally coded and stored on an electronic media so that it can be used by a computer.*

Other terms that one may come across are *digital resource* of *digital object*, both trying to stress that a document is more than text.

How does this relate to a *digital library* then? Is a digital library simply a collection of digital documents? No, it isn't. Of course, a digital library stores (generalized!) digital documents, but there (or should be) more to it. Again, several definitions are possible and in use. They did change over time, beginning with the view of a digital library as a collection of electronic information.

> [*Digital Library*] is the generic name for federated structures that provide humans both intellectual and physical access to the huge and growing worldwide networks of information encoded in multimedia digital formats.
> *(The University of Michigan Digital Library: This is Not Your Father's Library, Birmingham, MI, 1994)*

The following alternative concentrates more on the goal that one wants to achieve with a digital library.

> A *digital library* is a distributed technology environment which dramatically reduces barriers to the creation, dissemination, manipulation, storage, integration, and reuse of information by individuals and groups.
> *(Edward A. Fox, ed., Source Book on Digital Libraries, page 65 [Fox93])*

In the later sections of this thesis the definition below will be used. It gives a rather technical view but contains all important aspects.

Definition 3 *A* digital library *is a virtual collection of generalized digital documents, along with methods for access and retrieval, and for selection, organization, and maintenance of the collection.*

As can be seen, a digital library is much more than a simple collection of digital documents. It additionally contains techniques, rules and applications for usage and maintenance. It is important that the definition includes no terms like "location"' or "place", instead "virtual" is used. Of course, the documents in a digital library can be stored on one file-server at a specific place (e.g. a library), but more and more often they are distributed all over the world (a so-called *distributed digital library*) and connected through the Internet. Some people even say "the Internet is a digital library", but I do not agree: the methods for access and maintenance that work in the complete Internet are missing (yet?).

In the paragraphs above the term *information* has been used several times, so let us shortly discuss it. In computer science, the following definition is usually used.

Definition 4 Information *is a message that can be interpreted.*

This means, information is a 'higher form' of data. Not any arbitrary data, but a sequence of data that someone or something can understand. Of course, this border between data and information is often subjective and very difficult to define generally (e.g. `01100000` looks like random data to you and me, but for a processor it might be the command to return from a subroutine). Even more powerful is *knowledge*, which is a special kind of information. An information which a person gets, can become part of his or her knowledge. From the online dictionary, used already at the beginning of this section:

> [*Knowledge* is] the acquaintance with facts, truths, or principles, as from study or investigation.

2.1.2 Structure & Presentation

In the last section, the differences between data, information and knowledge were described. As knowledge is beyond the scope of this thesis, we are now going to discuss the two possible ways, to store information: either their presentation or their structure can be "frozen" into a data format.

Both representations have their advantages and disadvantages. As always, it depends on several factors which one to choose. Generally, a structural representation is considered to be more powerful, because it is always possible to derive an adequate presentation. The other way round is more difficult (see below and in Chapter 3), yet there are situations, where the presentation alone is considered sufficient (e.g. for a final output document which is only to be printed or only to be read by a customer).

Document Presentation

The term *presentation* as used here means the physical image (e.g. the layout of printed documents or the sound, if talking about audio data etc.) of information. For example the information *We will meet on Tuesday.* can be presented using red color in a specific font, with a size of 12 points starting at a position of 3 cm from the left and 2 cm from the top of a page... Following [FE00] the definition below will be used throughout this thesis:

Definition 5 *The physical form of a document considering the chosen media is called* presentation*. This can also be regarded as the* physical structure *of the document (opposing its information structure).*

As can immediately be seen, presentation is media dependent or depending on the output device. The layout defined in the example above will produce a good readable result when output by an color inkjet-printer on a white A4 page. The information would be unreadable (at least from an appropriate

distance) when printed onto a poster with several square meters size. The instructions would even fail completely, if an output media is addressed, which has no knowledge of color (e.g. a b/w laserprinter) or a two-dimensional position (e.g. a news-ticker). Additionally, presentation is receiver dependent. Physical impressions are always subjective. For example the color red usually indicates "Attention!", "Danger!" or "Stop!" for a person from a western society. However, it might express fun or happiness for someone from another culture. Or think about a blind person. The terms "red" or "sans serif" will have no meaning for him or her at all. The information would have to be converted to another media so that it can be perceived with a different sense than seeing.

These dependencies are some of the major drawbacks of a presentation-centered representation of information. Especially in the field of Digital Documents and Computer Science, there is another one: it is very difficult (and often impossible) to process information which is only available through its presentation by a machine, i.e. automatically. If a software wants to extract the time of the meeting from the layout of the example above, it first has to digitize the color values, has to recognize the characters, must recombine them to words and finally has to find a word that represents a time. Soft- and hardware for this task exists of course (i.e. scanners, OCR programs...), but the job is difficult, error-prone (e.g. an OCR program with a typical error ratio of 95% is considered very good, but that means, that several words in one line of text can be wrong!) and slow.

It must not be neglected, that a physical presentation of information bears some advantages. When at some point a specific information should be read a human being on a printout, it must be layouted. We are used to read black letters from a white paper and we can deduce structural information (like "this is the title of the document") from the optical impression at once very easily ("a picture says more than thousand words"). In a way, physical layout serves as an interface from the machine to the outside world (i.e. us).

Another advantage has become more or less irrelevant over the last few years: the performance issue. Converting structured documents to layouted representations whenever necessary (called on-the-fly, e.g. when someone wants to read the document) might take a lot of resources, possibly too much to be useful. For example, only ten years ago, ordinary PCs just were not fast enough to render a TEX-document (structure) to the screen (presentation) within fractions of a second. Today, this is no problem.

Document Structure

The structure of information or a document can be understood as its logical order. The usual example is a book which consists of chapters, sections, paragraphs and words. Such a structure is often hierarchical but this is no necessary requirement. Put in the form of a definition (again following [FE00]):

Definition 6 *The* structure representation *of a document is the order of relevant content elements. Often this is also called the* logical structure *of a document.*

As opposed to the physical presentation, such a structure is independent from a media or the form of output (at least it can and should be), because the "What" is specified instead of the "How". When necessary, it can easily (and automatically!) be mapped to any physical form. For example, the structural element `chapter` can be mapped to the following layout instructions when printed on paper: start a new page, change the headerline and set the title in a big font. Especially it is possible to map one structure to several different output layouts, as requested by the publisher or the receiver of the document. `chapter` could be converted to "leave some space, put a horizontal line, output the title in red" when the information is to be displayed in a Web browser as opposed to the piece of paper in the example above.

This is very powerful during a modern publication pipeline, where a book is to be published in several ways: in classical print, on the WWW, on a CD-ROM etc. (b.t.w. this is called *cross-media publishing*). It is not necessary to write three books, instead this diversion is only important in the last

step of the workflow, when the structural representation of the document is converted to three different layouts automatically (through a so called *stylesheet*, see Section 2.1.5).

Several other important advantages of structured documents are already known from the fields of Programming and Software Engineering: reuse, division of work and maintainability. As with a program source code, which by the way definitely is a digital document, or complex systems in general only well structured representations make these three tasks possible or at least easier. If several authors create documents using a structured format, then a new document can simply be assembled by putting the parts together. No general layout guidelines would have to be defined before and checked afterwards. Even a part from another, older document could be easily inserted and reused.

Last but not least, structural representations allow better searching. If the structure is defined properly (see also Section 2.1.6) one can exploit semantic information in a query. For example one can then ask for a document containing the term `Kohl` as the name of a person, instead of looking for all documents that use this term somewhere (which in this special case might result in a lot of German recipes).

As a conclusion, it should be requested that for digital documents information is to be represented by its structure whenever possible, except for the last step of a publication chain when the document is printed, displayed or converted to sound waves etc. This final presentation of the information (e.g. their layout, as far as text-oriented documents are concerned) may then be stored for efficiently reusing it later (i.e. caching). Therefore, from this section's point of view, a document format which can store the structural form of data is to be recommended for any intermediate step in a digital document workflow.

From the last paragraphs, it should also be clear that different kinds of software are necessary, to either work with the presentation of a document or to work with its structure and to generate a layout in the final step. See Section 2.2 for a further discussion of this fact. The modern WYSIWYG[2]-software usually being used today for creating text documents centers on the layout aspect (as is already implied by the term "See" in WYSIWYG). Therefore, most writers are conditioned to think of "big, bold, underlined" instead of "headline". It will need a good time of explanation, teaching and of course attractive software to change this. For example, publishing houses, acknowledging the advantages of structured documents, have finally started to follow this road.

2.1.3 Textformats

When it finally comes to the formats to store digital documents in, there are a lot of possibilities on how to categorize them. Because so many different formats exist and each day new ones are invented while others are no longer used, such a grouping is necessary to give a short overview. Of course not every document format will be mentioned, but I have chosen the most important ones which can serve as examples for the different categories.

Of course, one possible and sensible way to group document file formats is by the media type they represent. This approach will be followed here. This gets difficult when it comes to formats which allow to store several media types in one file (so called *multimedia formats*), but even then, one type is often the predominant. For example, most textformats can *also* store embedded images. As a result from the previous section (2.1.2) it is important whether a document format contains a structural representation or the physical presentation of the document. It is obvious, that this difference should also be used for a categorization. Thus the currently most popular structural document format *XML* will be introduced separate from the other formats (see Section 2.1.5).

Additionally it might be useful to distinguish between proprietary (i.e. most often closed source, having to be licensed) and open (i.e. open source, free to use) document formats. For each format described it will be mentioned to which of both groups it belongs. Often a separation between formats for storage and formats for transmission has been made. Generally this is no longer necessary, due to our modern high performance networks and the possibility to transmit 8 bit data. An exception to this rule

[2] "What You See Is What You Get"

is the field of so-called *progressive* or *interleaved* transmission of documents, which allows to open a document already before its transmission is finished. This feature is well-known from the WWW, where it is used for *JPEG* [ISO90] or *GIF* [Com87] images, but also for text documents stored as *PDF* [BC93].

We have seen in Section 2.1.1 that digital documents are much more than printed books in electronic form, but still most of them are text documents. Therefore the list of available formats is especially long in this category. And it makes a good point to start.

ASCII

The most often used (and perhaps least known) textformat might be *ASCII* (the *American Standard Code for Information Interchange*) in the original form defined by ISO 646 [ISO91]. ASCII, which strictly speaking is only a font encoding, is a 7 bit format, so it allows only 128 different characters (i.e. letters, capitals, digits and a few special characters like "#" or "&"). This limited range makes it insufficient for most languages besides English. Therefore a lot of national variants exist, which is a major drawback. For example the character sequence `schöne Grüße` might appear as `sch|ne Gr}~e` when the reader changes from a German to an American machine.

This code was soon extended to 8 bit characters, e.g. in form of the well known ISO 8859 [ISO98], that consists again of several flavors (ISO8859-1 = Latin-1 = West European, -2 = East European, ..., -6 = Arabic ...) but at least each of them is well-defined.

The big diversity of ASCII-formats has finally been overcome with the invention of *Unicode* [The91], which is a fully defined multi-byte format with 2^16 characters for most World languages. Several encodings exist to represent these characters as byte sequences. For example, *UTF-8* maps the ASCII-characters to their original (i.e. 7 bit) byte values, whereas other characters are represented by multi-byte codes. *UTF-16* simply uses 16 bit (= 2 byte) for each character.

The usual file-extension for text documents in ASCII format (or one of its descendants) is `.txt` (or sometimes `.asc`). As this format does not contain any structural or layout information, documents are simply represented as character-strings of arbitrary length. Finally, all ASCII formats could be considered as open formats, because the specifications are available and they can be used freely.

Proprietary Formats

Many other textformats are proprietary. In general, every word processing package has its own binary format to store text documents with embedded images and metadata (see Section 2.1.6) etc. Examples are *Microsoft Word* (file-extension `.doc`), *Star Office: Star Writer* (`.sdw`), *Adobe Framemaker* (`.fm`) and others.

The internal structures are often secret or simply not documented and their specification usually changes between different versions. The representation of the document content is rather presentation-oriented, some programs add more or less structural information. Additionally, user preferences, undo buffers and other program internal data are often included. Thus, there is not much to say about these formats here.

The most important disadvantage of these formats (especially for further processing of documents or for document interchange) should be obvious: the program which generated the file in the first place is needed in order to access it. This problem is somewhat moderated by the fact, that many programs include so-called *import/export modules* that allow the reading or writing of other programs' formats. But often the used conversion routines can only retain part of the information embedded in the file or they fail to work with a new version of the file format. Of course, these problems might be considered irrelevant for closed user groups (e.g. employees of a company within an intranet) that work with identical hard- and software.

As a conclusion, these formats are no good basis for a document workflow in an heterogeneous environment (like the Internet), although the current mainstream trend focuses on them.

```
{\rtf1\ansi\deff0
{\fonttbl{\f0\froman\fprq2\fcharset0 Times;}}
{\colortbl\red0\green0\blue0;\red255\green255\blue255;
\red128\green128\blue128;}
{\stylesheet{\s1\snext1 Standard;}
}
{\info{\comment StarWriter}{\vern5690}}\deftab720
{\*\pgdsctbl
{\pgdsc0\pgdscuse195\pgwsxn11905\pghsxn16837\marglsxn1800\margrsxn1800
\margtsxn1440\margbsxn1440\pgdscnxt0 Standard;}}
\paperh16837\paperw11905\margl1800\margr1800\margt1440\margb1440\sectd
\sbknone\pgwsxn11905\pghsxn16837\marglsxn1800\margrsxn1800\margtsxn1440
\margbsxn1440\ftnbj\ftnstart1\ftnrstcont\ftnnar\aenddoc\aftnrstcont
\aftnstart1\aftnnrlc
\pard\plain \s1 Title
\par
\par This is a rather senseless text...
\par
\par }
```

Figure 2.1: example of a simple document, created with *Star Office* and saved as RTF

RTF

A somewhat special case is the *Rich Text Format* (*RTF*) [Mic99] developed by Microsoft and originally used as an additional format for *Microsoft Word*. RTF is well-defined and the specification is freely available, therefore the ability to read and write this format has been added to most available word processing software and it has become a widely used format for text exchange (the least common denominator, one might say).

An RTF-file (with file-extension `.rtf`) uses only printable characters and consists of an hierarchical structure of (human-readable) commands, groups, tables and data. It can contain layout as well as structural elements and metadata, somehow resembling TEX-source (see below). It even includes character and paragraph layout templates (called stylesheets). Many of the (structural) elements are optional, meaning from the point of view of someone who wants to further process such documents it is possible to create good or bad RTF. Figure 2.1 shows an example of a simple document saved in RTF.

TEX, LATEX

TEX and especially *LATEX* are very different and a big step away from the proprietary, presentation-oriented document formats. To be precise, TEX is not just a format but a structured type-setting programming language. Nevertheless it is used to write text documents, to store them and, of course, to automatically generate layouted documents of high quality, ready for printing or for display.

TEX [Knu86] was developed and implemented by Donald Knuth in the early 1980s. Technically, the system consists of a compiler (`virtex`) that converts a document description (the source document, file-extension `.tex`) into a device independent layout description (*DVI*, file-extension `.dvi`), a complete programming language (including variables, conditional clauses, functions...), several macro-packages and a set of font descriptions. A DVI file can be previewed or printed, but often it is further converted into *PostScript* and/or *PDF* (see below).

The TEX system is very stable, it is freely available (incl. source-code) and it has become some sort of a standard in many research institutions and societies, not only because of its ability to set mathematical formulae in very high quality. However, because the original TEX (or *plain TEX*, as it is called) is rather

```
\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
   \maketitle
   Hello world!
\end{document}
```

Figure 2.2: An example for a structured document in LaTeX.

complex, it is difficult to learn and it is even harder to produce good results.

This situation improved when Leslie Lamport developed his macro package LaTeX in 1985 [Lam85]. Lamport created format templates for different kinds of documents (i.e. book, report, article, letter) and combined many low-level commands to create new powerful functions. Both was a major achievement towards the idea of a structural representation of information. With LaTeX the author does not have to care about the presentation of his document (the TeX-compiler will generate high quality layout) and thus he can concentrate on the structure. A simple example (from the WWW-pages of the LaTeX3 project, see [The00]) is displayed in Figure 2.2. In plain English this reads as:

- This document is an article.
- Its title is "Cartesian closed categories and the price of eggs".
- Its author is "Jane Doe".
- It was written in September 1994.
- The document consists of a title followed by the text "Hello world!"

As can easily be seen, this is nearly a perfect structural representation of the information contained in this document. This thesis itself is a good example for the layout that LaTeX will generate automatically from such descriptions.

Until today, the TeX-typesetting system has further evolved (LaTeX 2ε is the latest version, LaTeX3 is being developed [The00]) and several useful tools and packages have been added: BibTeX for the management of bibliography databases, `makeindex` for the generation of indexes and `hyperref` for the automatic creation of hyperlinks, to name a few. Even WYSIWYG-editors (like *Scientific Workplace* or *LyX*) and interfaces to computing algebra systems (like *Mathematica* or *Maple*) are available now.

PostScript, PDF

As opposed to LaTeX above, *Adobe PostScript* [Ado98] (*PS*) clearly is a layout format. It was developed in 1982 to exactly describe the layout of pages for printing in a vector-oriented way. PostScript includes geometric primitives, colors, fonts, raster images and 2D-transformations, but it is also a turing-complete programming language.

The PS format (file-extension `.ps`) is owned by *Adobe*, but the specifications are freely available and free implementations exist (e.g. *GhostScript*[3]). This format has become something like a standard for high quality output of text and graphics for professional printing and publishing, many b/w- and color printers use it as their default input language.

The *Portable Document Format* (*PDF*, file-extension `.pdf`) is a derivative of PostScript, also by *Adobe* [BC93]. It was developed as a layout format for WWW-environments and low resolution displays (like computer screens). It supports most of the primitives and features of PostScript, but it is no longer

[3] http://www.cs.wisc.edu/~ghost

turing-complete. The ability to emulate missing fonts, compression algorithms, hyperlinks, security mechanisms and lot of other features which are useful for digital documents have been added. This makes PDF currently the most important format for digital documents in heterogeneous environments, where high quality presentation is required.

Adobe provides a software (i.e. *Adobe Acrobat Reader*) to display or print PDF documents on many systems free of charge. Other third party viewers (many of them Freeware) exist (like *GhostScript*). The easiest way to create PDF is to convert PostScript using *Adobe Acrobat Distiller*, which is a commercial package. Again, free software to achieve similar results exists (e.g. PDF can be generated directly from TEX-source, using `pdftex`). Today, PDF more and more serves as a basis for workflows of digital documents, for which it is very well suited because of its interactive components, embedded annotations and digital signatures, access rights as well as security mechanisms.

Several of the modern *Markup Languages* like *SGML*, *HTML* and *XML* are also used as a format to store text documents, but because of their high relevance they will be discussed separated in Section 2.1.5.

2.1.4 Formats for Non-textual Documents

Some of the proprietary or presentation-oriented textformats mentioned in the previous section can also be used to store non-textual documents (e.g. images embedded in text), but special formats for the different media-types exist, of course. Of these, several important ones will be discussed in the following paragraphs, ordered by the usual categorization by media-types.

Raster Graphics

The term *raster graphics* is commonly used for two-dimensional images at a specific display resolution, represented by a rectangular field of *pixels* with digital color values at a specific color resolution (i.e. no. of colors or no. of different values per color component). This is clearly a presentation-oriented representation of information. A lot of different formats for raster graphics exist, many of them proprietary and many free. Usually they can be distinguished by the color resolution they are able to represent and by the ability to compress the image data. As far as compression is concerned, it can be further distinguished whether *lossless* or *lossy* compression schemes are used (or both). *Lossy* compression means that a high compression ratio can be achieved, at the cost of not being able to reproduce the original information exactly (though possibly only with an invisible error).

GIF *GIF*, the *Graphics Interchange Format* [Com87] has been created by *CompuServe*. It supports color tables with up to 256 entries (with 8 bit-resolution for the color components red, green and blue each, so called *Truecolor*) and lossless compression (i.e. based on *LZW* [Wel84]). Additionally, an *interlaced*-mode is supported, that allows to display the complete image in a lower resolution before it is completely transmitted. The file-extension is `.gif`.

GIF was originally available free of charge, but since some years a license to create GIF images is required. Nevertheless, it is still the most used format on the WWW, especially for small graphics or images with a low number of colors.

BMP The *MS Windows Device Independent Bitmap* (short *BMP*, file-extension `.bmp`) is the default image format of *Microsoft Windows*. It supports color tables (≤ 256 colors) or truecolor pixels and an inefficient lossless compression (i.e. a form of *RunLength-Encoding*). It is a very simple, freely available and openly specified format.

TIFF *TIFF*, short for *Tag(ged) Image File Format* (file-extension `.tif`) by *Aldus Corp.* [Ald92] is a very versatile image format which supports many different data-types and optional metadata,

organized in tagged blocks. It offers color resolutions of 1 – 48 bits/pixel as well as several compression algorithms. The specification is openly available and free implementations exist (e.g. `libtiff`[4]).

TIFF is some kind of a standard for the transfer of large, high resolution images (e.g. from digital scanners). Because of the wide variety of optional data-types supported, it is often not possible to read a specific TIFF, although the used software claims to be able to "read TIFF".

PNG The newly developed Open Source graphics file format *PNG* (pronounced "ping" for *Portable Network Graphics*, file-extension `.png`) [Bou97] supports most of the options that also TIFF, BMP and GIF offer. It was explicitly developed to be able to replace many existing formats. Features which are especially useful on the WWW like progressive image display and CRCs against transmission errors have been added. As of now, support for this format is widely available, but not always stable.

JPEG Strictly speaking, there is no standard named *JPEG* but everybody refers to an image compression standard from the *Joint Photographic Experts Group* [ISO90] (i.e. *JFIF* for *JPEG File Interchange Format*) by that name. JPEG is a file format (file-extension `.jpeg`) to store truecolor or grayscale raster images using a lossy compression algorithm with arbitrary quality. A *Discrete-Cosine Transformation* (DCT) is used on blocks of 8×8 pixels and after that, the number of the resulting components is reduced, as required by the selected quality. The resulting data is then further compressed using lossless techniques (like LZW), usually reaching a reduction in file-size of about a factor of ten. Additionally, JPEG supports progressive transmission and display. The specification is available and many Freeware implementations exist.

As the used lossy compression scheme works fairly well for photographic images, it has become a standard in that area (e.g. for digital cameras).

Vector Graphics

A *vector graphics* file format describes the content of the resulting image by placing geometric primitives and using 2D transformations (see below for 3D). It is thus (or at least can be) device/media independent and so makes a step towards a more structure oriented representation of the information.

There are several application dependent formats, mostly those of the widely used CAD software (like *DXF* for *AutoCAD*) or *CorelDraw* for example, but no standard which is widely used today. Except maybe *CGM* (the *Computer Graphics Metafile* [ISO99c]) which is accepted by wide parts of the industry. Instead, the following two which are very popular will be mentioned:

PostScript As already mentioned in the section on textformats (see 2.1.3), PostScript [Ado98] is a vector page description format being mostly used to output text. It can also be used to describe two-dimensional vector graphics application-domain independently and is either read or written by most software. Due to its turing-completeness, full-featured import filters are hard to implement and often they are not stable or sufficient enough, to parse complex documents.

SVG *SVG* or *Scalable Vector Graphics* (file-extension `.svg`) as developed by the *W3C SVG Working Group* [W3C01a] is an XML-application (see Section 2.1.5) and thus inherits all features of XML like structural representation, the easy ability to process and to check validity as well as modularization. It has been designed to be a language describing two-dimensional vector graphics for WWW-applications and for data exchange. Additionally it supports embedded text and raster images. SVG resembles PostScript as well as CGM in many aspects. Because the complete specification is openly available and free as well as commercial implementations are available, SVG

[4] http://www.libtiff.org

has the potential to become a widely used standard, at least for the application fields that it was designed for.

3D Graphics

3D graphics is very similar to vector graphics in the previous paragraphs, but here three-dimensional objects are placed at 3D-positions and 3D-transformations are used to define a scene. Complex objects are either described as a combination of basic objects or by their surfaces, which are often approximated by a mesh of triangles. Additionally, most 3D graphics formats store surface properties, light sources and a virtual camera. The documents (in the broader sense from Section 2.1.1) stored in these formats can then be displayed interactively by a viewer or they can be used as input for photorealistic visualization.

Again, each of the available software packages (called *modeling software*) has its own file format. There exist two (or, strictly speaking, only one) software and platform independent standards:

VRML *VRML*, the *Virtual Reality Modeling Language* (file-extension `.wrl`) is an ISO-standardized format [The98] which describes the contents of a scene as a structural representation in readable text. Additional features like animation, interactive elements and hyperlinks are also included in the latest version. The specification as well as several implementations are freely available and most relevant software can at least read this format.

X3D *X3D* (for *Extensible 3D*, file-extension `.x3d`), currently being developed by the *Web 3D Consortium* is strictly speaking not a new format. Instead, it is the successor to VRML and an XML-application (again, inheriting all advantages of this meta-format).

Audio

Generally, *audio formats* are similar to raster graphics. In both cases, an analog signal is sampled to digital values at a specific frequency. The different formats can be categorized by the way how they store these values and how the data is compressed. Often there is no compression applied at all (e.g. on CD Digital Audio), but also lossless compression and very elaborate lossy compression schemes are available. Other differences include how many parallel audio streams (so called *channels*) can be stored and which metadata (see Section 2.1.6) is embedded.

WAVE *WAVE*, originally from *Microsoft*, is a very simple format (file-extension `.wav`) that stores the values from the sampling process described above. It supports up to two channels (i.e. stereo) and a resolution of up to 16 bit per sample. Because of its simplicity and its free specification, this format is very common and serves as least common denominator for the exchange of audio data. As no data compression is included, WAVE quickly generates large files.

Real The proprietary *Real*-format (or *RealAudio* to be precise, because also a video format with this name exists) is a proprietary format from *RealNetworks* (file-extension `.ra`, besides others). It is often used in Internet-environments for the *streaming* of audio information, meaning that data is sent continuously while being received as well as played simultaneously. Real supports many different audio formats and compression schemes. Player-software is available for many systems free of charge, but professional tools for commercial applications have to be licensed.

MP3 *MP3* (correctly *Motion Picture Experts Group, Audio Layer 3*, file-extension `.mp3`) [ISO99b], developed by a *Fraunhofer Institute*, uses a lossy compression exploiting the (limited) abilities of the human ear. It supports several sampling frequencies as well as sample resolutions and *bitrates* (bit/s, the input signal is compressed to this output quality). Streaming is also possible. Free implementations for encoders and decoders exist, but the original ones have to be licensed.

This format is currently very popular, as it allows decent (called CD-like) quality at low bitrates (e.g. 128 KBit/s $\approx$ 1 MByte/min compared to $\approx$ 10 MByte/min on CD Digital Audio).

MIDI The *Musical Instrument Digital Interface* (or short *MIDI*) is not only an audio format but also a set of protocol and electrical specifications for the connections between musical instruments and computers. The audio format-part of MIDI is different from the previous examples, because it describes notes, scores and features of the sound to be played. The original signal is not sampled, instead a structural description on how to synthesize it is given. The resulting files are very compact (a few bytes per note to be played) and free implementations exist.

Video

The first *video formats* simply stored a video document as a consecutive sequence of raster images. Each of the raster graphics formats mentioned above can be used for this. The problem is the huge amount of data that will be generated. It even gets worse, because usually also audio information (possibly several channels) is stored together with the video. For example, if one stores a TV recording (*PAL* standard) as a stream of uncompressed images, $\approx$ 30 MByte of data per second (704×576 pixels, 24 bit color resolution, 25 frames/s) would be generated. For most application scenarios this is definitely to much. Even if one of the good, lossy compression algorithms for images (e.g. that of JPEG) were applied, still about 3 MByte/s would be written (which would give only 25 minutes of video on a DVD, and no audio yet). The solution is to use the coherence between consecutive images (usually, they will look very similar) and to compress groups of pictures (*GOP*s) instead of single images.

Something else is important for video formats: many of them are so-called *envelope formats*, meaning they do not specify the internal structure nor the compression scheme of embedded video and audio streams but leave this to the so-called *codecs* (a pair of encoding- and decoding-algorithms). One video format does then support several codecs. *AVI* and *QuickTime* are well-known examples of such envelope formats (see below).

QuickTime The proprietary *QuickTime* file format from *Apple* [App00] is one of the envelope formats mentioned above (file-extension `.mov`, besides others). It supports a wide range of audio and video codecs, from simple to very efficient ones. Additionally, interactive features are supported (e.g. a user can click on a specific area of the movie display) and the streaming of movies of possible. The specification of the file format itself is freely available, but most codecs have to be licensed. Quicktime is the default video format for *Apple Macintosh* computers. *Microsoft Windows* is also supported by a free player application from Apple. Most Quicktime videos cannot be played on other systems, because of the missing codecs.

AVI The *Audio Video Interleaved* format (*AVI*, file-extension `.avi`) from *Microsoft* is also an envelope format for a lot of audio and video codecs, many of these being available as Freeware. Even codecs for some MPEG-formats (see below) are available. Additionally, a streaming extension (*ASF*, *Active Streaming Format*) has been developed by Microsoft. As AVI is the default format of *MS Windows* it has become some sort of a standard together with this operating system. This format currently is very popular and can be played as well as created on most systems.

Real *Real* or *RealVideo* is the equivalent to RealAudio from *RealNetworks* (file-extension `.rm` and others). It is a proprietary format for the storage and especially for the streaming of combined video and audio information, often at low bitrates. Several proprietary codecs are included. Simple player applications are available for free supporting many systems, commercial encoders have to be licensed though.

MPEG *MPEG* (file-extension `.mpg`) is used as the name for several video and audio compression algorithms and file formats from the *Motion Picture Experts Group* which are ISO standards [ISO99a].

Currently *MPEG-1*, *MPEG-2* and *MPEG-4* are used to store video documents (incl. audio). The compression algorithms are similar to JPEG image compression, but additionally the coherence between consecutive video frames is exploited. The formats are very strict as far as the supported image and color resolutions are concerned (usually matching established TV or digital video standards). MPEG-1 is used for computer movies and so-called *Video CDs*, where near VHS-video quality is achieved with a bitrate of 1150 kBit/s. MPEG-2 is the format for movies on *Digital Versatile Discs* (*DVD*) and MPEG-4 could become the standard for the upcoming *High Definition Television* (*HDTV*). The big advantage of MPEG videos is, that they can be played on almost all computer systems as well as on consumer DVD-players.

As a conclusion of the previous sections on document formats, we now want to give some recommendations on what format to use, depending on the media type of the document and its intended further usage.

For text documents, that are final versions (i.e. to be given to other people for reading or printing only) PDF should be the first choice. For documents that will be modified and further processed RTF or LaTeX might be an option, but one should definitely have a look at the modern *markup languages* presented in the next section.

Raster graphics should be stored as PNG or JPEG (when higher, lossy compression is required). PNG is modern, clearly specified, openly available and has the potential as well as the features to replace all other formats with lossless compression.

As far as vector graphics documents are concerned, currently only PostScript can be recommended. SVG might become the future standard, but currently its support is not sufficient and stable enough yet.

If one wants to store audio files, the MP3-standard should be considered or WAVE if higher quality (i.e. lossless compression) is required. Both are well supported and openly specified. Under certain circumstances, MIDI might be the most efficient solution.

Nearly the same applies for videos. Here, the different MPEG-standards offer high quality at acceptable bitrates using well defined standards and algorithms. Additionally they can be and certainly are implemented in hardware. Other available formats with similar (or sometimes even superior) quality would bind the user to certain manufacturers and/or platforms. Of course, if one wants to further process a video document at the highest possible quality, a format which offers lossless compression (like AVI using an appropriate codec) should be used.

2.1.5 Extensible Markup Language

As discussed in Section 2.1.2 a structural representation of information is desirable. Several formats that allow this (at least in part) have been introduced already, but the current explosion of the availability and the usage of so-called *markup languages* offers a lot of new, profitable opportunities (technically as well as economically). But let's start by going one step back.

It is obviously necessary to separate the information from its structure and from the possible presentation. This can be achieved by a layered approach (three layers in this case):

1. the pure information itself, i.e. *data*
2. the logical *structure* of this information
3. one *presentation* (out of several possible) for each structural element

These layers can be implemented by embedding the structural information within the information (because it is unique) by the use of special commands or tags (the so-called *markup*). This structured document is then converted to a presentation-oriented form in a second step by adding style (through a

so-called *stylesheet*). And this is what *markup languages* are all about: a way to embed structural information within the document. This basic idea is not particularly new (e.g. the old UNIX text-formatting tool `roff` already used this approach), but especially through the WWW-inspired *Extensible Markup Language* (*XML*, see below) it has come to new life. XML and the well-known HTML are both based on SGML. Therefore these languages will be discussed in a reversed order.

SGML

It all began with the creation of *SGML* (the *Standard Generalized Markup Language*) which is based on the *Generalized Markup Language* (*GML*, see [Gol90]) and implements a logical or generic markup. SGML became an ISO standard in 1986 [ISO89]. Strictly speaking, SGML is not a markup language but a grammar to define an application specific markup language that uses SGML syntax. With SGML the hierarchical structure (and in part the semantics) of a whole class of documents is defined using a *DTD* (short for *Document Type Definition*). The language (or format) of such a document class is called an *SGML-application*. Software to read and interpret all documents of a class (and to check them against the appropriate DTD) is named *SGML*-parser. Usually, a general parser is used which is initialized with the DTD and which then understands the specific document class.

SGML is very complex and it is not easy to implement general parsers. Therefore, it is not widely used except for some large companies (e.g. airplane manufacturers). Nevertheless, also a language to define stylesheets for SGML-documents called *DSSSL* (*Document Style Semantics and Specification Language*, [ISO96]) is available.

HTML

The markup language that most people are acquainted with today is the *Hypertext Markup Language* (*HTML*) [W3C99a]. HTML is an SGML-application (an SGML-DTD exists). It was developed by Tim Berners-Lee in 1989 while inventing what was later called the *World Wide Web* (which, by the way, is nothing more than HTML as a document format, HTTP as transmission protocol and a Browser for navigation and display) [BLF99].

An HTML-document consists mainly of text, but additionally hyperlinks and references to images are included. Originally, HTML offered a limited set of structural elements (called *tags*) like `title`, `heading`, `list` etc.) and also several tags describing layout (like `underlined` or `bold`). In short time with the spreading of the WWW the requirements grew. Therefore, later versions of HTML were extended to allow more complex presentation-oriented features, like colors, absolute positioning and embedded multimedia objects. Also, the ability to create interactive forms and to embed scripting languages was added. Today, even tags that only work with one specific browser (and thus are outside of the specification, of course) are in use. On the WWW a lot of documents are not valid according to the latest HTML standard (but most browsers will display them “somehow”).

Generally speaking, the idea of building well-defined structural representations of information has been lost. It is still possible to create such documents of course. With the help of the *Cascading Stylesheet*-language (*CSS*) [W3C98] it is then possible to map a structured HTML-document to a specific presentation. This is (more or less) supported by modern browsers, but only rarely used (besides creating even fancier layouts with the help of a CSS). See Figure 2.3 for the HTML-code of a simple, structured document using some embedded CSS-statements for styling.

In the future, HTML should be replaced by *XHTML* which is an XML-application and again a well-defined structural representation, that requires a stylesheet (CSS or *XSL*, see below) to define the layout.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
  <HEAD>
    <TITLE>HTML</TITLE>
    <STYLE type="text/css">
      EM { color: blue }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>HTML</H1>
    <P>Is a <EM>markup language</EM>.
  </BODY>
</HTML>
```

Figure 2.3: example of a simple structured HTML-document, using an embedded CSS stylesheet

XML

The *Extensible Markup Language* (*XML*) has been created by the World Wide Web Consortium (W3C) in 1998 [W3C00b] with the intention to create a "lightweight SGML for the Internet". As the specification puts it, the goal was:

> [...] to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

XML is a subset of SGML (meaning every valid XML-document is a valid SGML-document). One of its design goals was to have only 10 % of SGML's complexity but to get 90 % of the functionality. As a result, several rarely used constructs of SGML were left out and the syntax of XML was defined more strict, to make the documents easier to parse. As in SGML it is possible to define the structure of a class of documents by a Document Type Definition. But in XML, this is not necessary. A document, that only follows the syntax requirements of XML is already considered a so-called *well-defined* XML-document. In this case, there is no set of elements predefined. An arbitrary set of tags (resp. those needed for the specific application) can simply be used and any XML-parser will be able to read the document. This is the meaning of the 'Extensible' in XML. When additionally a Document Type Definition is provided, and a document fulfills the given rules, it is considered a *valid* XML-document according to this specific DTD. This validity is checked by so-called *validating parsers*.

The design goals of XML have undoubtedly been achieved. This is also proven by the big success of this markup language in the Internet community and beyond. Today, XML is everywhere: it is understood by Web-browsers, it is used by applications and databases as well as to create new document formats. See Figure 2.4 for an example of a simple, well-defined XML-document.

By now, a lot of languages, applications and formats based on XML have been standardized. The following list gives a short (and incomplete) overview of these XML-applications:

XSL *XSL*, the *Extensible Stylesheet Language* [W3C01c], is an XML application that allows to convert XML documents to layout-oriented representations by transforming the structure (*XSLT*, *XSL-Transformations*) into another XML document describing the layout (*XSL-FO*, *XSL-Formatting Objects*) or into an HTML or another text document. As an alternative, CSS can be used to add presentation information to an XML-document.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- XML example -->

<MyDocument>
  <Chapter label="C1">
    <Heading>Introduction</Heading>
    <Content Language="english">
      <Par>Hello World!</Par>
      <Par>How are <emph>you</emph> doing?</Par>
    </Content>
  </Chapter>
</MyDocument>
```

Figure 2.4: example of a simple well-defined XML-document

XHTML the *Extensible Hypertext Markup Language* (*XHTML*) [W3C00a] is a reformulation of HTML 4 in XML. The goal is to smoothly replace HTML as the default language for text documents on the WWW.

SVG *SVG*, the *Scalable Vector Graphics* [W3C01a], is an XML-application to describe vector graphics. See Section 2.1.4 above.

X3D the *Extensible 3D* (*X3D*) XML-application will become the successor to the 3D graphics format VRML, see in Section 2.1.4.

XML Schema the XML-application *XML Schema* [W3C01b] is another way (instead of using a DTD) to describe the syntax of a class of XML-documents. But it goes far beyond, because XML-Schema allows also to define semantics. Non only the structure but also the acceptable content of valid documents can be specified. For example, it can be requested that a tag `Number` contains only a sequence of digits using an object-oriented framework of types, classes and inheritance.

RDF *RDF*, the *Resource Description Framework* [W3C99b], is an XML-application that allows to describe a set of *metadata* plus a model defining the metadata elements. See Section 2.1.6 below.

2.1.6 Metadata

One of the key issues that increases the power and the value of digital documents (e.g. in contrast to printed materials) is the availability of *metadata*. Especially, metadata enables or at least improves the automated processing of documents like indexing, searching or converting because it can add some sort of semantics to the pure data. A human being knows the meaning of words intuitively, but a machine does not (at least not until today...). So the computer needs help for this task, and that is where metadata (and structure, see above) comes into play.

Put shortly, metadata is simply "data about data" or more technically "data which supports operations carried out on information objects". A similar definition is given in [FE00]:

Definition 7 Metadata *are data describing a document, also called descriptors. Usually, these are bibliographic data.*

Well known examples are the information (metadata) in a library catalog about the available publications (data) or the access rights (metadata) of files (data) stored in a file-system. Also other documents like program source code (author, comments, copyright...) and newspapers (date, editor, price...) usually contain metadata. An interesting point is, that of course metadata is also data and thus, it can have

Title	Subject	Description	Creator	Publisher
Contributor	Date	Type	Format	Identifier
Source	Language	Relation	Coverage	Rights

Table 2.1: The 15 core elements (descriptors) of the Dublin Core metadata standard.

metadata (which is *not* called meta-metadata). For example, a list of prices (metadata) can have an expiration date, which is metadata about metadata. What happens often, is that one document's data is another document's metadata and the other way round.

Metadata and structural representations of information make almost a perfect match: the structure splits and orders the information in independent pieces and metadata assigns these additional semantic information (i.e. a meaning). What is needed is therefore a standard way to bind metadata to structural elements. This is available for documents formats like HTML and XML (see below).

It is obvious from the examples above, that the term *metadata* is new but the basic idea is not. Many disciplines use long established standards for their metadata, especially the library sciences. Examples for these are *MARC* (for *Machine Readable Cataloging*, from the US Library of Congress [Fur98]) and *MAB* (for *Maschinelles Austauschformat für Bibliotheken* used by German libraries [Die92]). As can be seen, even within one discipline the standards often differ from country to country. This problem gets even worse on the Internet, where people of different sciences from all over the world want to exchange digital documents. Actually, two standards are needed for this: one standard for the metadata itself and one standard on how to combine the metadata with the data (i.e. the documents). We need to mention the following two developments:

Dublin Core

Dublin Core (named after Dublin, Ohio, where one of the creators, the Online Computer Library Center – OCLC – is located) [WKLW98] is the standard for metadata of digital documents that is currently the most accepted on the Internet. It is coordinated and extended by the *Dublin Core Metadata Initiative* (*DCMI*) [DCM02]. Dublin Core (or *DC*) is set of only 15 descriptors or categories. These small number of *core* elements (see Table 2.1) – compared to several hundred of other standards like MARC – were explicitly chosen to span all disciplines and topics. They are simple enough, so that they can be filled by the authors themselves and at the same time the descriptors are powerful enough to be sufficient for most applications. They were designed to enable a simple and semantically compatible exchange of metadata for digital documents. The syntax for the metadata or ways on how to embed it into the documents described were left open.

Later, further (optional) refinements and syntactical schemes were defined. These allow for example to explicitly specify which date is given (e.g. `Date.Created`, `Date.Modified`) or in which format the date is given. For further information, see the WWW-homepage of the DCMI [DCM02].

How the Dublin Core-metadata is connected to the digital document described, depends on the format of the document itself. Usually a *namespace* `DC` is used to indicate the type of metadata. The descriptors as well as the values are embedded using features of the document format. Two examples for documents stored as HTML and as XML are shown in Figure 2.5. For HTML the `META`-tag is used and for XML elements from the XML-namespace `dc` are inserted respectively.

RDF

RDF, the *Resource Description Format* [W3C99b, Mil98], is not a standard for metadata like Dublin Core, but a "meta-model for metadata". It is being developed by the Metadata Working Group of the

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
  <HEAD>
    <META name="DC.Title" content="Metadata">
    <META name="DC.Creator" content="Marco Zens">
    <META name="DC.Date.created" scheme="W3C-DTF"
          content="2002-04-18T09:55-01:00">
[...]

<?xml version="1.0" encoding="iso8859-1"?>
<MyDocument>
  <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:title>Metadata</dc:title>
    <dc:creator>Marco Zens</dc:creator>
    <dc:date refinement="created" scheme="W3C-DTF">
      2002-04-18T09:55-01:00
    </dc:date>
  </metadata>
[...]
```

Figure 2.5: A simple example on how to embed Dublin Core metadata into HTML and XML documents.

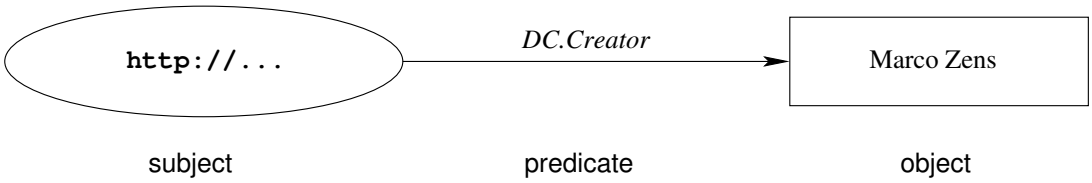


Figure 2.6: This is the graph for a simple RDF-statement, following the RDF Model Specification.

World Wide Web-Consortium (W3C) and now a part of the broader Semantic Web Activity. RDF currently consists of two parts: the *RDF Model and Syntax Specification* and the *RDF Schema Specification*. Shortly, these are a model and a language to describe the metadata as well as a scheme to describe its semantics. Used together, these would allow the automatic parsing and interpretation of arbitrary metadata.

The *RDF Model* allows the generalized, syntax-independent description of metadata through directed graphs. It consists of *resources* (objects, identified by an *Uniform Resource Identifier*, *URI*), *literals* (character strings) and *predicates* or *properties* (names to describe a resource). These are joined in a graph to form *statements* containing a subject (a resource), a predicate (a resource) and an object (a resource or a literal). See Figure 2.6 for an example of such a statement. A statement is again a resource, which allows statements about statements (called *reification*). Additionally, several more complex *structures* like *bag*, *sequence* and *alternative* are defined.

A graph built of these elements can then be stored as an XML-document using *RDF Syntax*. See Figure 2.7 for the XML-file describing the graph from Figure 2.6.

An *RDF Schema* finally, is an RDF-graph (resp. an XML-document in RDF Syntax) that defines resources (metadata) and their properties, including semantic restrictions like a range of values. To achieve this, predefined types as well as object-oriented mechanisms like classes and inheritance are available.

```
<?xml version="1.0" encoding="iso8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description about="http://www.eg.org/index.html">
    <dc:Creator>Marco Zens</dc:Creator>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.7: A simple RDF statement described in the RDF XML-syntax.

As important it is to use metadata at all, it is even more important to use a standardized form to be able to process the document and the metadata automatically and efficiently. Dublin Core gives a good start, because it is accepted world-wide, especially on the Internet. Other, more specialized or more accurate metadata schemes can be formally specified in machine-readable form by the use of RDF. Every producer of digital documents should make use of it.

2.2 Creation of Digital Documents

Before a digital document can be stored in a specific format (see Section 2.1) and before it can be accessed (or 'read', see Section 2.3) it has to be created first. If one wants to create a "good" electronic document (i.e. choose an appropriate format, use a structural representation, add metadata etc., see the previous sections), a document with *electronic added value* (see Section 2.2.3), this is a non-trivial task. One has to come up with an usable document structure. A software to create the document based on this structure easily and comfortably has to be available. The author needs support for creating and adding the metadata. And, last but not least, the document has to be converted to one or often several presentation-oriented representations in the final step of the publication pipeline (*cross-media publishing*). Even after the publication, quality assurance is necessary (e.g. maintaining a list of errata, publishing updates etc.). Keep in mind, we are not talking (solely) about text documents here, instead we have to look at many different media-types.

In the following sections, we will take a closer look at many of these aspects. Before we start with this, let's add one other point: how does a digital library fit into these scenarios? The answer is "barely". The creation of digital documents is not the core task of digital libraries. It is a major section of the research field on DLs, though. Possible exceptions are the enhancement of existing documents with additional value (see below) or the re-publication on another media.

2.2.1 Retro-Digitization

One possible way to create digital documents is called *retro-digitization*. This means, that existing printed, recorded, filmed, painted etc. (i.e. analog) documents are converted to a digital format. We must not forget that by far the major part of all existing documents (including those from the past) are not available in digital form. The technical devices to achieve this range from simple scanners over OCR-software (*optical character recognition*, the original text, i.e. a sequence of characters, is regenerated from the layout image) to sophisticated, highly specialized machines that automatically turn over the pages of a book, that is being scanned. Other examples include so-called *3D scanners* that use laser- or *structured light*-technologies to digitize the form and often even the material of three-dimensional objects. Many readers will have already done a simple retro-digitization by themselves. For example, they might have put a filled out form on their scanner for archiving or they possibly have grabbed a TV programme with the computer for later viewing.

So, the main question is not whether retro-digitization is possible, but whether it is useful or necessary. Several reasons in favor of retro-digitization are often mentioned, like the following:

- The additional availability of a document in digital form, can make it attractive to new customers and prepare it for new markets. The electronic added value that a digital document can offer (see Section 2.2.3) is a powerful advertising point. As a result, often *economical reasons* for the retro-digitization of existing documents can be found.

- Many groups, institutions (including libraries) and national organizations are concerned with the *preservation of cultural heritage*. The retro-digitization of a valuable old book or other artifacts is one way to make security backups (besides the still unsolved problems of the long-time preservation of digital materials, see Section 2.3.2).

- Closely tied to the previous reason is the idea of making old, valuable or sensible documents or objects (like those stored in museums) *publicly available* without risking damage or even theft. This can be easily achieved using retro-digitization. The object can then be shown interactively on a simple display.

  Another important application of this is the education of the population in poor countries. It is most possibly cheaper to distribute a CD-ROM (and a device for reading it, of course) to every small village, then to build a conventional library (with books) at each of them.

- Last but not least, one reason for retro-digitization is simply the wish to access all materials for a specific topic in the same manner, i.e. electronically in this case. For example, readers that can access the last three volumes of a scientific journal easily online, might want this also for the other volumes, *without a change of media*.

As can be seen, there are many good reasons for retro-digitization and practice shows, that they are valid. For example, currently in Germany a joint retro-digitization project of several libraries is funded, which tries to make available the title pages of 300.000 prints from the 17th century (*VD17* [DHW97]). Similar projects can be found world-wide, see [GMS+98] where several are described. Examples for the retro-digitization of journal volumes can be easily found within many scientific societies like the *ACM*[5] or the *Eurographics Association*[6].

The most cited reasons against retro-digitization are that it is difficult and that it is expensive. The enormous cost is mostly a result of the difficulty to convert existing documents to high-quality digital ones. But of course, it depends on both, the resources and the quality requirements.

As an example, let's consider someone has digitized one page of a book. At a scan resolution of 150 dpi (which is enough for display and printing) a colored page of size A4 would result in approx. 6.4 MByte of uncompressed data. Now just think of thousands of books with hundreds of pages. This results in a huge amount of data. Additionally, this is only image data, meaning it does not contain letters and words and it cannot be used for searching or for further processing. To allow such additional features and to reduce the size, *optical character recognition*-algorithms would have to be applied. Good OCR-software achieves recognition ratios of about 98 %. This sounds good, but it means that on one page of 4000 characters 80 could be wrong, e.g. the first digit of a number! If this is important, another pass using spelling verification and even a manual check is necessary. At this point, the new digital document is still not structured and it does not have any metadata attached. These additional steps of work sum up to the high cost mentioned.

To add another number to the discussion: in a journal article [Odl99] Odlyzko estimates, that the complete literature in mathematics of the past could be retro-digitized (scanning only) for about 18

[5] http://portal.acm.org/dl.cfm
[6] http://diglib.eg.org

million US-$. This sounds much, but ten times this amount is currently spent yearly for *new* mathematical journals!

As a conclusion, retro-digitization is a possible way to create digital documents. There are good reasons for and against it, but as the practice shows there are cases, environments or tasks which justify this effort.

2.2.2 Creation of New Documents

Several years ago, a new document was created in the mind of the author and then layouted using a typewriter, a pen or by applying paint to the wall of a cave. With the invention of the computer and especially the personal computer this has changed in many ways. Today, it is even possible, that a digital document is never published to any physical form (e.g. printed). In the following paragraphs, we will concentrate on text documents because the techniques to handle these are the most advanced and most statements can easily be transferred to other media. One must not forget though, that text documents are only one aspect of generalized documents (see 2.1.1).

Generally, there are two options to create a new document (besides retro-digitization, see Section 2.2.1): automatically or by an author. Automatic creation of a document means, that a software uses a fixed pattern, an algorithm or heuristics to assemble a new document from already existing pieces of information. Examples are a report generated from a database, the results of a query on a WWW search-engine or the short summary of a scientific paper, which was generated using sophisticated techniques (see for example [SABS94]). The first procedure is known since long time, the second works more or less well and the last example is possible due to the latest results of the research field called *Information Retrieval* (e.g. [SM83]). Information Retrieval is a very challenging subject and plays a major role in Digital Library-research, but we won't be able to discuss it in more detail here.

From the point of view of a publication pipeline with well-structured digital documents, the automatic creation of documents is not especially challenging. It can be exactly specified what structure and what format to use (one only has to decide upon an appropriate format, see below) without problems, because a machine then executes the task. The remaining issues are the same as for the other class of new documents, generated by authors.

The group of people, who create documents can be further divided into skilled and unskilled authors. One just needs to tell the former simply what is needed (e.g. "a press-optimized PDF, generated using our LaTeX-template") and they will deliver it, having found and used the appropriate tool. The latter need clear guidelines and, even more important, a set of tools that helps them to easily fulfill the requirements. Usually, authors are not very eager to spend hours on reading a documentation. Instead they want to concentrate on what their primary job is: to produce good content (i.e. information). And that is what they should be doing, isn't it?

So what authors really need to generate "good" (i.e. well-structured etc.) digital documents is a predefined structure, a standardized format and (most important) a tool and/or template that supports them to achieve this as efficiently as possible. An even better solution would be, if this tool was their favorite document editor (e.g. a word processor) which they use every day.

In an ideal world, authors would use a platform-independent, easy-to-use word processor, that generated documents in a well-defined structural representation. Unfortunately, this is not available (yet?). The software to generate text documents that is in use today, can be divided into three groups:

- *presentation oriented* software, that follows the WYSIWYG-paradigm (like *Microsoft Word*),
- *content oriented* tools, like LaTeX that foster the author to specify *what* he wants to express instead of *how*, and

- *structure oriented* editors which let authors work on the structural representation of their document explicitly (e.g. *Adobe FrameMaker+SGML*).

Sadly but true, most authors prefer Microsoft Word today (besides a few research communities where LaTeX is still the tool of choice) and often they even are not aware of the resulting problems. However, as discussed above, it would be no good idea to *force* authors to use other software.

There are several approaches to overcome this situation. In any case, a document template and guidelines are needed, to enable authors to build the best possible Word document. The point is that Word is not as bad as its reputation. It has features like named paragraph- and character-format templates, interactive text fields and scripting-language controlled GUI-elements. These allow to at least somewhat separate structure and content from the presentation; the authors only have to be taught how and *why* to use them.

Many research projects (and resulting applications) concentrate on the idea to rebuild a sensible structure from the available presentation-oriented document at a later stage, i.e. after the author has completed his work. This approach will play a major role in my discussion of the research project *GoldenGate* in Chapter 3. During this project we tried to silently embed invisible hints into a Word document by using paragraph- and character-format templates with a visual appearance that is identical to the normal text. Such a document was then in several steps converted into an XML document, changing the embedded hints into elements (i.e. tags). The Master's Thesis of Carsten Oberscheid [Obe99] describes this process in more detail. The participants of the German *Dissertations Online*-project [7] use a similar solution. Here, a Word-document based on a given template is converted to a specific SGML/XML-DTD (*DiML* for *Dissertation Markup Language*) using several scripts and macros. As a third example, I want to mention scientific publishers (like *Springer*), who have also started to convert Word documents to XML/SGML-format, in order to be able to use these for cross-media publishing (like in their online digital library, see e.g. *Springer LINK*[8]). In general they use the Word document in RTF and apply several scripts, conversion steps and manual work to achieve this.

Another approach is to enrich Word documents with structure information while they are being created/edited, either (semi-) automatically or by encouraging the author to do it. Usually this means, that a plug-in (an *AddIn*, as Microsoft calls it) or a package of *VBA* macros (*Visual Basic for Applications*, the scripting language of *MS Office*) is installed, which in cooperation with a given document template controls the structure of any new document created by the author or even creates a complete SGML/XML-document.

Several research projects are working on this idea, one of these being the *aTool*-project of the "Sonderfördermaßnahme" *WEP* (*Werkzeuge für Elektronisches Publizieren*) in the *GlobalInfo*-program[9] of the German Federal Ministry for Research and Education (see also Section 5.3). *aTool* [FGKM00] is a plug-in for MS Word which tries to automatically map named formats directly to element types of an arbitrary XML-DTD. The generated document is then interactively checked against this DTD and corrected by the author.

Last but not least, another solution can be to offer specialized applications to initially generate a good Word document. The author can then insert missing data using Word or make small modifications. The initial document can be generated through a WWW-form, by an independent tool or after executing a VBA-macro (which could use interactive GUI-elements like a dialog to get necessary information from the author). The problem with this approach is that the author could (unintentionally) modify or even destroy some necessary structure elements afterwards. It is often tried to avoid this problem, by using the document protection-mechanism of MS Word which allows to protect certain parts of a document

[7] http://www.dissonline.de
[8] http://link.springer.de
[9] http://www.global-info.org

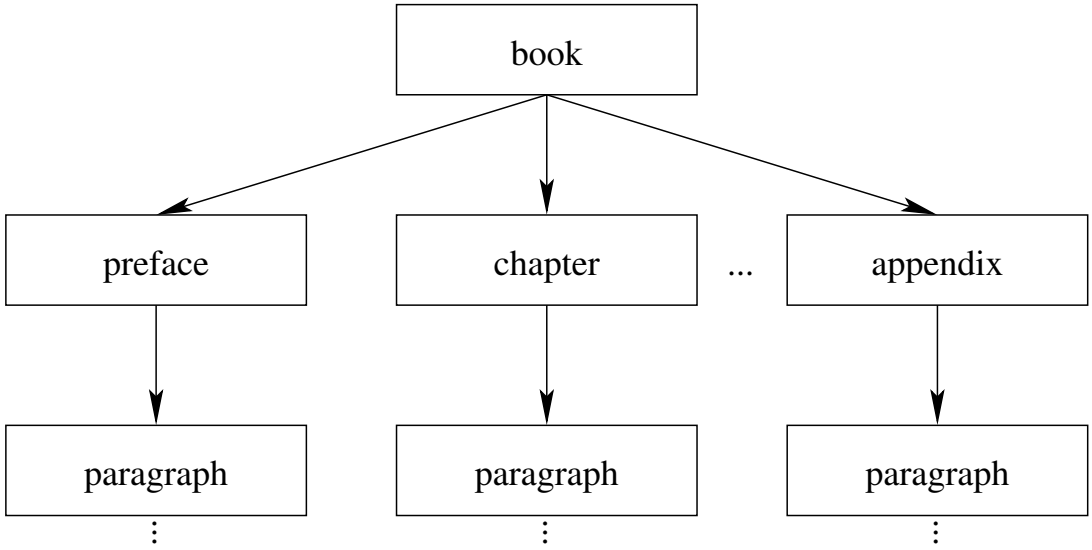


Figure 2.8: Example of a simple hierarchical structure of a book.

against modification. For example, the document template used for *GoldenGate* (see Chapter 3) makes heavy use of this approach.

Going one step further, let's say we get an appropriately well structured document from the author and can convert it into a structural representation. The question arises in which document format it should be stored and which structural elements it should contain. The answer to the first question is simple (see also Section 2.1): today, this will be SGML or XML. The latter is more difficult. Usual text documents contain some form of a hierarchical structure (see also Figure 2.8), a vertical ordering of document units: book – chapter – section – paragraph – word. It is definitely a good idea to use a standardized structure, or speaking in markup language-terms, a standardized *Document Type Definition* (*DTD*). A popular example is *DocBook* [WM99] which exists both as SGML- and XML-DTD and is intended as a structure particularly well suited to books and papers about computer hardware and software. See Figure 2.9 for the XML representation of the structure in Figure 2.8.

2.2.3 Electronic Added Value

As laid out in the beginning of this chapter (see 2.1.1), a digital document is (or should be) more than just a digital representation of for example a printed work. One difference is the structural representation, the other important ones are often called *electronic added value*. This added value is the sum of advantages that a digital document can offer the reader as opposed to an analog form. The following is a (incomplete) list:

- A digital document, especially a text document, can offer many efficient ways of *navigation*. A reader can directly access arbitrary pages, chapters, figures or indexes and return to where he was before. Also fine-grained bookmarks or unusual movements like up/down (in the hierarchy) are possible.
- Another very efficient way to navigate in digital documents is the use of *Hyperlinks*, i.e. shortcuts between different document units or to other documents that are easy to follow (usually by a click). These even lead to the new paradigm of information access called *browsing*: a reader can follow his very own path through information space, he does not have to follow a route predefined by someone else (i.e. the author) anymore.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
          "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
<book>
<bookinfo>
  <title>My First Book</title>
  <author>
    <firstname>Marco</firstname><surname>Zens</surname>
  </author>
</bookinfo>

<preface><title>Preface</title>
<para>This book shows...</para>
</preface>

<chapter><title>Chapter 1</title>
<para>In this chapter...</para>
</chapter>

<appendix><title>Appendix 1</title>
<para>Here is the source...</para>
</appendix>
</book>
```

Figure 2.9: A simple example of a document structure represented by the DocBook XML-DTD.

What is more important, an author or publisher can enrich his document by adding hyperlinks to secondary sources of information, like to reviews, to the homepage of the author/publisher, to extended examples and other additional material.

- As compared to printed documents, an electronic document usually allows to use *extended layout* and design features like colors, images and different fonts. Another interesting point is the possibility to dynamically adapt a document to the reader ("if you are new to this field, start with Chapter 1–3"). The danger to exaggerate these new options exists, though.

- The previous possibility can even be expanded to the use of many different media-types in one document, a so-called *multimedia* document. Difficult textual content can be clarified by the embedding of audio, animations or complete movies.

- The *securing of quality* is much more easier for a digital document because it can be easily updated or even replaced. By exchanging a document on the server, all readers of this instance will automatically and transparently receive the latest version. Even partial updates are no problem.

What is definitely missing from the above list is metadata (see Section 2.1.6). As far as the creation of documents is concerned, the situation with metadata is very similar to that of structure. It can be added manually or automatically either instantly or afterwards. For the manual creation, again comfortable tools and standardized formats are necessary. Often metadata is generated automatically during the document assembly process. For example, MS Word uses the first text that an author formats with a heading style as initial value for its internal title field. The necessary heuristics and algorithms are from the research fields of *Information Retrieval* [SM83] and *Data Mining* [WF00].

One example for the automatic generation of metadata is the so-called *Autonomous Citation Indexing* (*ACI*) [LGB99]. A popular implementation is the *NEC ResearchIndex*[10] (formerly known as *CiteSeer*). This service scans scientific documents which are available online and tries to extract bibliographic references. The goal is to generate a database of citations which can then be used for literature search and evaluation.

In this section we have tried to give an overview of the field of (text-) document creation. The final question which remains is about the role which the publishing houses play (or should play) during this process. Especially within some scientific communities this is changing dramatically, due to the new technical possibilities. The former tasks of a publisher were to support the author, to convert the document into a form usable for publication, to control the quality and finally to distribute the document through several channels (i.e. to publish it). Many of these steps could now be performed by the author on his own. He can create his new document in a standardized format like PDF and publish it on the WWW. In most scientific communities the quality control is done be the community members by themselves through *peer reviewing*. Therefore, today we can see online journals being made available without the involvement of a commercial publishing house or only in loose cooperation, like *J.UCS* (the *Journal of Universal Computer Science*[11]) for example. See also in Section 5.4.5.

There are some people who argue, that publishing houses are not needed anymore for scientific publications. I don't agree. As can be seen in the previous sections, (most) authors need a lot of support to create good documents. This should come from the publishers. And this is only the beginning of a complete publishing pipeline. Additionally, there are many ways to distribute a document, not only the WWW (which has itself a lot of problems like long time preservation, link consistence, searching & finding... see the next section below). Cross-media publishing is an important aspect and it is the publishing houses that have the expertise, the contacts and the economical power for this task. What is important, is that publishers not only raise their prices. They have to reconsider their role, present their strengths and adjust to the new situation and its requirements.

2.3 Access to Digital Documents

After the previous section about the creation of digital documents, I now want to have a look at the other end of a complete publication pipeline: the distribution (i.e. the actual publishing). This deals with the question of how, when and where a document is stored, how it can be found and how it is finally accessed.

Usually, a digital document is published in several different formats and instances (*cross-media publishing*), e.g. online, on a CD-ROM and as a conventional print. A CD-ROM and a printed publication can be distributed through the usual channels like libraries or by mail order. The necessary techniques and processes are established and well known, so they are not of interest in this thesis. We will concentrate now on the pure digital distribution channels, especially the online publication.

This is the point, where digital libraries come into consideration. Today, most digital documents are made available through the Internet or the WWW, meaning both are used as ways to access digital libraries and even as large digital libraries themselves. Besides, it has already been mentioned (see Section 2.1.1) that I do not consider the Internet/WWW a digital library. To be precise, the Internet is a network of loosely coupled computers together with a set of rules and protocols (like *TCP/IP*) that enable communication and data transfer. The World Wide Web is simply one specific application on this global network. Only the big popularity of the WWW today leads to the often made misinterpretation to identify the WWW with the Internet. It is this popularity (based on the simple but comfortable look & feel: browsing, clicking, multimedia) that offers many opportunities for digital documents. However, also many problems emerge.

[10] http://www.citeseer.com
[11] http://www.jucs.org

The good point is that every document can be accessed by nearly all people around the world in a matter of seconds by simply downloading it. At least this is technically possible. But how do these people know about this document in the first place? How do they find it? Or, does the author/the publisher want free access for everyone? For traditional (i.e. printed) publications the answer is simple and well known: possible readers can go to a library, check the *catalog* and read the book. Or they can identify the book in question by its unique global identifier (*ISBN* for *International Standard Book Number*[12]) and order it in a book shop. It is the task of digital libraries to implement and offer processes based on these metaphors for digital documents. But, as the global digital library is not existing yet we have to cope with the situation by using the methods offered on the Internet.

But having all documents stored in a digital library with a catalog still is not sufficient. The reason is, that there are many digital libraries around the world, each with their own indexes. Thus, one won't be able to find a document which is only listed in the catalog of another library. It is immediately clear what is needed: a (bibliographic) metadata standard and an exchange protocol so that different catalogs can interact or so that they can even be joined. Such standards are available and they have been mentioned already in Section 2.1.6. What I like to add here is the availability of a new, open standard on this subject, the *Open Archive Initiative's*[13] *protocol for metadata harvesting* [LV01, RF01].

Once the exact location of a document is known, the access is easy. Possible access restrictions set aside, which would then require methods of identification, authentification and payment which for themselves are very challenging, but they won't be considered here. The document location can be specified for example by an *URL* (*Uniform Resource Locator*, consisting of a protocol, a hostname and a resource identifier) or as combination of an Internet address and a filename. A reader can then transfer the document to his own workstation and read it, given that the document is stored in a standardized format (see Section 2.1). Thus, the major problem is to find the document, to get to know its exact location (e.g. its URL).

2.3.1 Searching (and Finding)

Searching is one of the basic operations in computer science, and as such many sophisticated algorithms have been developed (and are being developed). We will concentrate here on methods and requirements to find a (generalized) digital document. Generally, the goal of *searching* is to find one element of a set which matches a specific criterion. In other words, we want to find one (or several) digital document within a collection (or digital library) which matches our requirements, specified by a query. Such a query is often formulated with a query language (like *SQL*, the *Structured Query Language* [DD93]) but for digital documents and especially for text documents easier ways to explain what a user wants are available (see below). When no perfect match can be found, often several close matches are delivered as result of a query. These results are then ordered according to their (possible) relevance, which is called *ranking*. The key requirements to enable efficient searching are the ability to create an index (i.e. an ordered directory of query terms, each linked to the relevant documents) and to sort (i.e. to order following a specific criterion). Any unordered set of objects can only be searched by looking at each single object sequentially, which might take a long time (just think of thousands of documents with the size of a book).

The efficiency of a search algorithm for documents is measured using the two quality parameters *precision* and *recall*. Recall is defined as the number of documents relevant to the user that were delivered in the result divided by the complete number of relevant documents that are contained in the collection which has been searched. The second parameter, precision, is the part of the delivered documents that are really relevant to the user's query. This means that one parameter can always be increased at the cost of the other. For example, simply delivering all documents of the collection would yield a recall of 1 (or

[12] http://www.isbn-international.org
[13] http://www.openarchives.org

100 %) but in this case, the precision would be very small. Therefore it is always the goal to achieve high values for both parameters. Usually 90 % are considered very good.

There are several methods to improve a search algorithm, especially for text documents. These include linguistic functions like *truncation* (of prefixes or suffixes), *stemming* (removing conjugation or declination) or phonetic similarities. Other algorithms use semantic tools like *dictionaries*, *classification schemes* or *thesauri*. A third group of improvements are based on *relevance feedback*, where the user is asked to evaluate the presented results. The search algorithm can then adapt itself and (hopefully) deliver better results in a subsequent query. A final example of how the search in large document collections can be improved is called *Cross Language Information Retrieval* (*CLIR*) [Gre98]. It allows to search for documents in several different languages using only one query (in a single language).

A very often used algorithm to find a text document is the *full-text query*. Here, the user simply specifies one or several terms. The documents (from a set) that contain all these terms are delivered as result. This approach can be easily extended by also delivering those documents which contain only some of the queried terms. When applied, usually the precision of the result is reduced and the recall is increased. Additionally, the number of terms that were found in a particular document can be used as a simple measure for ranking. The next step is to use a *boolean query*, where boolean operators (`AND`, `OR`, `NOT`) are used to combine the query terms. Additional operators like `NEAR` (meaning that both operands need to be found "closely" together, e.g. on the same page) are very popular. Generally speaking, a boolean query *can* increase the precision without decreasing the recall as much as a simple query with many terms (which is most often implemented as an implicit boolean query using `AND`).

If metadata is available, the search can be further refined by only looking in particular metadata fields, like title and author. Sometimes only the metadata is available (instead of the full-text), especially when the documents are stored in several heterogeneous, presentation-oriented formats. Each specific search engine can only extract the full-text from a limited range of formats (the popular formats PDF and PostScript are difficult, for example). Of course, the possibility to search for metadata elements is a great opportunity because it is much more precise and also faster. But in this case, the user needs to know what he is looking for and an appropriate metadata field has to be available. If someone is looking for documents about "hierarchical radiosity" the names of the authors and the dates of the publication will not be of much help. Often even the title is not sufficient: a document titled "Photorealistic Visualization" most probably contains what the user is searching for but it will not be delivered as a result in this case.

A third way to find text documents is to query by example, for example a user can ask to "find all documents about a similar subject like *this* one". Of course, such subjective measures like "content similarity" are not easily transferred into algorithms. Nevertheless, several heuristics based on statistical measures and probability functions have been developed by researchers in the field of Information Retrieval over the last years. A simple example is the *vector space*-measure, where the set of (non-trivial) terms in all documents form a multi-dimensional vector space. Each document is then one point (or one vector) in this space, with the coordinates being the number of how many times each term is contained in the document. Finally, the distance of the points and/or the angle between the vectors forms a simple similarity measure that can be used to compute and to order the results of the query.

The first and the second approach (the latter in a limited manner) are available on the Internet as *search engines*, also called *crawlers* or *spiders* following the internal names of the first implementations. Well known examples are *Google*[14] and *AltaVista*[15]. Technically they start with some document, create a full-text index and/or an index based on metadata and then follow the hyperlinks embedded in the document to reach other resources. This allows to access more and more documents over time, without ever coming to an end. Many studies show that even the best search engines have not indexed more than about 30 % of the complete WWW. The most important reasons for this problem are that, first the WWW is growing to fast and second, many documents will not be reached because they are not linked

[14] http://www.google.de
[15] http://www.altavista.com

from others or because they are only generated dynamically after a sensible request which can not be emulated by a "stupid" algorithm. An extension of these services are called *meta search engines* like *metacrawler*[16]. They do not index documents themselves. Instead they forward a user's query to several conventional crawlers transparently and then merge the different results to produce a single list. This can definitely improve the recall over that of any single search engine. The major challenge to these algorithms are the merging of the results and especially their ranking as well as the elimination of double entries when several search engines report the same hits.

Most of all these concepts and problems discussed above are also valid when searching is extended from text documents to generalized documents of other media-types. There is only one major difference: usually the job gets a lot harder. For example it is easy to find a word in a text but it is already very difficult to find a simple graphical object like a circle in a raster image. So for other media-types like video and audio, most often either query by example (e.g. "find an image similar to this one") is applied or special characteristics are used to form a query (like "images containing much red" or "sound with high frequencies"). Nevertheless, some very good results are achieved. To just give a starting point two examples should be named. First, Clausen et al. developed techniques for automatic indexing and retrieval of score-based audio data [KC01] during the *MiDiLib*-project. Second, Salesin et al. worked with multiresolution wavelet decompositions to search in an image database using a query image which is similar to the intended target (*Fast Multiresolution Image Querying*) [JFS95].

It is obvious, that especially for non-textual documents metadata is very important to allow efficient and precise searching. Therefore the new MPEG-7[17] standard (currently still a Working Draft, see for example [Hun99]), the so-called *Multimedia Content Description Interface*, is very important and offers a lot of potential for the future.

An alternative to searching is always the use of an available hierarchy or a categorization. This can be created manually or automatically, both forms exist on the Internet and in every digital library. A manual categorization is very time consuming (and thus expensive) and sometimes subjective. The automatic approach is technically very challenging, but algorithms for the clustering of documents based on their content and the subject extraction have been developed. An example are the so-called *Self Organizing Maps* (*SOM*) [Koh95] which allow concept-based searching and show very promising results, especially when used on document collections containing one class or a small number of similar document classes. By the way, the task of sorting documents following their content becomes very easy when appropriate metadata added by the author or the publisher is already available (see also Section 2.2.3). This is especially true, when a standardized thesaurus (like, for example, the *ACM CCS*, the *Computing Classification Scheme* of the ACM[18]) is used for classification.

A well known example of such hierarchical directories are sites on the World Wide Web like *Yahoo*[19], which today are called *portals*. They serve not only as hierarchies to access documents on the Internet, often in connection with full-text search engines, but they also offer a lot of other services (like news, weather forecasts, travel information etc.).

2.3.2 Storage

Another question is, how the actual document is stored – generally within a digital library and particularly on the Internet. The most simple solution is to store digital documents in a *file-system* and to convert the URL to a filename with a *Web-server*. This approach is (still) often used, but it leads to several problems. First, file-systems do not offer any support neither for the search algorithms described in the previous section nor for metadata. Second, and most important, such URLs that contain filenames are instable.

[16] http://www.metacrawler.com
[17] http://ipsi.fhg.de/delite/Projects/MPEG7
[18] http://www.acm.org/class
[19] http://www.yahoo.com

Whenever the file is renamed, moved or deleted, the URL becomes invalid (and thus any hyperlinks using it). Those missing links (and the corresponding HTTP-error `404`) are a well-known problem. Analyses have shown for example that approx. 20 % of all hyperlinks contained in scientific papers become invalid during the first year after publication [LCG+00].

A solution is to use a storage system that is generally described as *Document Management System* or *Content Management System*[20]. This is the name for a wide range of software systems from a simple file-system with indexing facilities up to full-fledged *Information-*, *Knowledge-* or *Workflow Management Systems* which use *Database Management Systems* as their document storage. These kinds of software concentrate on different aspects of document management but they all offer sophisticated features for storage, searching and access control. Shortly, *Information/Knowledge Management Systems* support users to work with the information contained in documents, like building references automatically, offering advanced query algorithms or combining existing knowledge to find answers to new problems. Examples are the *Hyperwave eKnowledge Suite*[21] or the *Fulcrum KnowledgeServer* from *Hummingbird*[22]. *Workflow Management Systems* like the *IBM MQSeries* or *COSA Workflow* from *Ley*[23], tend to stress more the support for electronic workflows. They help to model workflow processes and to enable access to the necessary documents for specified actions and actors. In most cases, general *Database Management Systems* (*DMS*) like *Oracle9i* or *IBM's DB2* are used to store and to find the documents, additional data and references between these resources. The idea of generalized digital documents (of many different media-types) is especially supported by so-called *Multimedia-DMS* which are able to manage arbitrary document formats using *BLOBS* (for *Binary Large Objects*).

The problem of invalid links can be solved by the application of an additional level of abstraction. This means, that a hyperlink does not reference a document itself directly (i.e. a location) but a virtual resource (i.e. an identifier) (also called *URI* for *Uniform Resource Identifier*) instead. This identifier is then mapped to a physical location in a second step, either by a simple table within a document management system or by a special service of an external authority. As a result, the physical location of a document (i.e. its filename or its position in a hierarchy) can be changed and afterwards only the mapping table between identifier and location has to be updated. All references (i.e. hyperlinks) within other documents do not have to be changed and remain valid. This is similar to finding a book within a library by its *signature* instead of its position on a specific bookshelf. Standards which have been developed for this approach are the *Digital Object Identifier* (*DOI*) [Pas99] and the *Uniform Resource Name* (*URN*) [SM94].

Additionally, many document management systems treat hyperlinks themselves (here: most often only internal links, but sometimes also external ones) as first-hand objects (i.e. documents) within their database. This has the advantage that all sophisticated features (like searching, access, version control...) are available and also it enables so-called *bidirectional links*. These allow to propagate additional information or modifications from a linked document to the objects that hold hyperlinks to this specific resource. Bidirectional links even make it very easy to efficiently implement queries for all documents that reference another one.

We need to consider here another problem of the storage of digital documents: long time preservation. This is not about the ability to find a document after a long time (see above) but to be able to read (or see, hear, interpret etc.) it. This is a problem including hardware (e.g. "How to read an 9 inch floppy disk?") and software (like "Which program can import a *WordStar 3* file?"). It is possible to read hieroglyphics that were carved in a stone wall thousands of years ago, but how many of us can still load a letter which we have written on our first personal computer? Two solutions to this problems are thinkable and currently in use: either a digital document must be archived together with the necessary hardware and

[20] http://www.contentmanager.de
[21] http://www.hyperwave.com
[22] http://www.hummingbird.com
[23] http://www.ley.com

software to process it or the document has to be converted again and again to a contemporary medium and file-format every few years. Both approaches require substantial amounts of time, money and physical storage capacity (e.g. for the old hardware). At least the conversion can be made a lot easier, if well-defined and structured document formats are used (like for example XML, see Section 2.1).

We should not forget that besides the storage of the final work in a digital library and the traditional (meaning paper-based) publication, there are other ways to distribute digital documents, which have to be mentioned shortly: *preprint* publication and *print-on-demand*. An example for the former is the well-known preprint server *arXiv.org e-Print archive*[24], originally located in Los Alamos as a digital library of preprints on high-energy physics. Such a server makes digital documents available which are not yet "officially" published (i.e. printed, in a book or journal) or which will never be distributed by a publisher. The former are often removed again later on.

Because it is often more convenient to have a work available in printed form, print-on-demand allows to make exactly one physical copy from the requested document whenever needed. This was made possible by the invention of printing machines which allow to produce single complete books at high quality automatically for a low price. Today, already many digital libraries and scientific societies offer this service for their digital documents.

2.4 Summary

During this chapter on digital documents many definitions, concepts and techniques were introduced and discussed which form the framework and the foundation for the applications and the projects described in the following chapters.

We started with the definitions of a *generalized digital document* as an electronic, self-contained unit of information and a *digital library* as a virtual collection of these, including methods for access and maintenance (see Section 2.1.1). Such an information unit can be stored in form describing the *presentation* i.e. the physical form, or as a *structure* representation which defines the logical structure, i.e. the order of relevant content elements. The latter is independent from a media or the form of output and allows for more efficient reuse, division of work, maintenance and searching. Therefore, a digital document should be in form of a structure representation whenever possible (see in Section 2.1.2).

Digital documents can be stored in many different formats. These can be easily grouped according to the supported media-types but they also differ in whether they are machine or human readable, whether they contain structure or presentation and whether they are freely available or proprietary. Amongst the well known formats for text documents are *ASCII*, *RTF*, LaTeX, *PostScript* and *PDF* as well as the proprietary formats of many word-processing systems (like *Microsoft Word*). They were discussed in Section 2.1.3. Formats for non-textual documents (see Section 2.1.4) include those for raster graphics (like *GIF*, *BMP*, *TIFF*, *PNG* or *JPEG*), for vector graphics (e.g. *PostScript* or *SVG*) and for 3D graphics (*VRML/X3D*). Other media types like audio (with the formats *WAVE*, *Real*, *MP3* and *MIDI*) as well as video (*QuickTime*, *AVI*, *Real* and *MPEG*) were also presented. An important conclusion is that generally the best format does not exist, only one which is most suited under given circumstances.

For the storage of a structural representation, so-called *markup languages* (which embed structural information with the actual data in form of elements or tags) are very useful and currently very successful (see Section 2.1.5). Presentation information is assigned to structural elements (identified by their markup) with the help of *stylesheets*. *XML* and *HTML* are both based on *SGML* which implements a generic markup and is a grammar to define an application specific markup language (with SGML-syntax), called an *SGML-application*. The structure of a whole class of documents is defined by a *DTD*. *HTML*, the format for text documents on the WWW, is an example for such an SGML-application, meaning a HTML-DTD exists. *XML*, developed as a lightweight SGML for the Internet, is a subset of SGML

[24] http://arXiv.org

with many complex features removed. An XML document is considered *valid*, when it conforms to all syntax requirements and it is called *well-formed* when it fulfills the rules of a DTD. For both, HTML and XML, layout can be added by *CSS*, but for XML documents *XSL* (being an XML-application itself) is better suited.

The automatic processing of digital documents (i.e. indexing, searching, etc.) is greatly improved if *metadata* (being information about data) is available. See the definition in Section 2.1.6. Metadata can add semantic information to structural elements (e.g. within a markup language). For this to work, both, a metadata standard and a standard way to bind metadata to structure elements is needed. *Dublin Core* is the former, consisting of 15 descriptors (i.e. categories) which are on one hand simple enough to be easily usable and on the other hand powerful enough to be useful. Another important standard is *RDF*, which is a model and a language to describe metadata as well as a scheme to describe the metadata semantics. When used in this way, RDF can enable the automatic parsing and interpretation of arbitrary metadata.

Section 2.2 discussed the techniques and the problems to create good digital documents, i.e. documents with *electronic added value*, meaning with efficient navigation, hyperlinks, extended layout and multimedia (see also Section 2.2.3). Software to create the document, its structure as well as the metadata is necessary. The conversion to one or several presentation forms (called *cross-media publishing*), quality assurance and the handling of many different media-types are other important issues.

One way to create digital documents is by *retro-digitization* (see Section 2.2.1), i.e. the scanning and the digitization of "analog" documents. Retro-digitization is difficult and expensive but it is possible and it can be useful because of economical reasons, for the preservation of cultural heritage, to make valuable documents publicly available without risk or to avoid a change of media within an information set.

New digital documents can be created automatically (e.g. a report generated from a database) or by authors (either skilled or unskilled). As described in Section 2.2.2, ideally authors should use platform-independent word-processors to generate documents in a well-defined structural representation. However, in the real world, they use *MS Word* (or some of them LaTeX) to produce already layouted documents. In any case, authors should be provided document templates and compact guidelines. Other possible approaches include the rebuilding of a structure from the available presentation after the author has completed the document or the enrichment of *Word* documents with structure information while they are being edited.

Once the document is ready, the other end of the publication pipeline (i.e. the distribution) becomes interesting. Section 2.3 discussed pure digital distribution channels, which is especially the online publication on the Internet. There, every document can be accessed by nearly all people around the world in a matter of seconds, if they can find it and if they are allowed to. As far as searching and access control are concerned, it is one task of digital libraries to offer processes based on the well-known metaphors for traditional publications. An example would be an exchange protocol and standard for metadata to be able to combine different, distributed catalogs.

Searching, discussed in Section 2.3.1, is one of the basic and therefore most investigated operations in Computer Science. Sophisticated methods to query for text documents are available. Their quality is measured using the parameters *precision* and *recall*. The results (or their presentation) are often improved by using *full-text queries*, *boolean queries*, *ranking*, available metadata or through *query by example*. The latter is especially useful for non-textual media-types which are very difficult to handle. On the Internet, so-called *search engines* who try to traverse the complete information space and implement one or several of the techniques mentioned above, are very popular. A final possibility, is of course always the creation of information hierarchies or categorizations (either manually or automatically) to ease the process of searching and (hopefully) finding.

Finally, the storage of digital documents on the Internet could be simply handled by a *Web server* who maps *URLs* to file-names (see in Section 2.3.2). This is not very powerful and leads to missing links (whenever a file is renamed, moved or deleted). Therefore, *Document Management Systems* which

use real *databases* for storage and implement an additional abstraction layer between document identifier (e.g. a *DOI* or an *URN*) and physical location should be used. Often also *bi-directional links* are supported as first-class DB elements. Another important issue is the long-time preservation of digital materials. This can only be achieved by either archiving also the hard- and software necessary for access or by converting the document formats to newer ones every few years. Both tasks are easier if well-defined, structural document formats are used.

Chapter 3

GoldenGate

This chapter deals with the research project *GoldenGate*, which focuses on the electronic submission, the managing and the approval of grant proposals at the German Research Foundation, based on standard Internet and office tools. It is the first of three application scenarios (see also the following Chapters 4 and 5) about the modern use of digital documents. In the following sections a short introduction to the specific problem which we solved using a new system design (see Section 3.2) will be given. After that we will discuss the prototype implementation as well as the extensions which became necessary (3.3). This immediately leads to a description of the project results and the lessons which were learnt (see Section 3.4).

3.1 Introduction

The *Deutsche Forschungsgemeinschaft*[1] (shortly *DFG*, i.e. the German Research Foundation) is the largest non-commercial provider of research funding in Germany. They support several thousand projects as well as researchers from every field of scientific research and education with an annual volume of more than 1000 million Euro.

When the project *GoldenGate* [FZ00, FZ99a] was started, the DFG interacted with the outside world only by paper documents, and mostly this is still the case. Internally, electronic support was provided by an integrated, proprietary database management system (called "All-in-One", which neatly describes the system structure btw.) that could only handle simple documents in ASCII format (see Section 2.1.3). At this time all offices were already equipped with modern PCs running Microsoft Windows NT 4 as operating system. Therefore a VT220-terminal emulation software was used to access the in-house database.

The latter points alone make it clear that a modernization was necessary. The DFG themselves have acknowledged the need for a new, modern system which can handle generalized documents (especially multimedia) for grant proposals, reviews and internal documentation. It has to be open to external users (e.g. applicants and reviewers) and must be flexible enough to be used by all different departments, each being responsible for a different field of research. Additional requirements were, that a new system had to run on today's predominant office architecture (PCs running MS Windows), offering a modern graphical user interface (GUI) and, last but not least, implementing the latest achievements from the ongoing developments in the fields of workflow and document management.

Summing up, DFG were looking for a document and workflow management system to make the daily departmental work more efficient (e.g. by offering electronic submission of grant proposals and process-oriented workflows) and to make all relevant information available to all groups (i.e. applicants, reviewers and DFG offices).

[1] http://www.dfg.de

The issues raised in the previous paragraphs are the foundation for *GoldenGate*. It was started in 1997 as a cooperative research project of the German Research Foundation, Department for Computer Science (lead by Dr. A. Engelke), and the Digital Library Lab (the author of this thesis being a member) of the Computer Graphics Group[2] at the University of Bonn (now at the TU Braunschweig), with a running time of initially one year, later extended to three years. The goal of this project was the design and prototype development of a complete electronic workflow at the DFG. It should allow proposals for research grants to be submitted in electronic form by researchers and to be stored and managed (e.g. made available to reviewers) in a database until the final approval and of course over the full life-cycle of each project, including knowledge engineering related tasks executed well past the active period of individual research projects.

One problem was that the traditional work processes in the individual DFG departments which should be augmented by electronic workflows during the course of this project obviously were never formally defined in a complete and consistent way. Because we could not find one single person who was able to define all rules and formalisms, we had to develop our system using an iterative approach. On the other hand, the advantage of this situation was that we could freely choose an architecture which is powerful as well as flexible enough to be easily adapted to any new requirement in each iteration step (see the following Section 3.2).

In the following, the basic system design which we developed as well as the implementation of our prototype based on this ideas will be presented.

3.2 Electronic Workflow of Documents

With the advancement of digital documents, Digital Libraries (see Chapter 2), Desktop Publishing (DTP) and the availability of networked computers in almost every office, a workflow exclusively based on electronic documents has become a viable alternative for a continuously growing community. Information can be entered into a computer, forwarded to other people, reviewed, modified and finally stored, without ever using paper (as a transport medium) during this full life cycle. In a lot of situations this use of digital documents (as opposed to paper) bears a lot of advantages, for both the sender and the receiver of information.

Of course, paper has some important advantages over an electronic file, therefore simply replacing printed documents on a desktop by their electronic source in a file-system (e.g. on a virtual desktop that many operating systems or GUIs offer) alone cannot be a satisfying solution. Instead, those advantages have to be modeled as accurately as possible whenever attempting to replace workflows similar to the one described above by their electronic counterparts. I will further discuss these features of paper documents and how they can be emulated electronically in the following section.

3.2.1 Common Features of Information Management Systems

Imagine a piece of paper with some important information sitting on a desk. What is it, that this "analog" document inherently offers from the viewpoint of Information or Document Management (see also in Section 2.3.2)?

- First, this document can only be read (i.e. accessed for reading) by a person being present in that room. If the paper was locked away in a closed drawer, even this would be impossible (well, to a certain extent...). It is even more important that only a person which is actually holding the paper document in his or her hands can make changes to the document content or add own comments (i.e. write-access).

[2] http://graphics.tu-bs.de

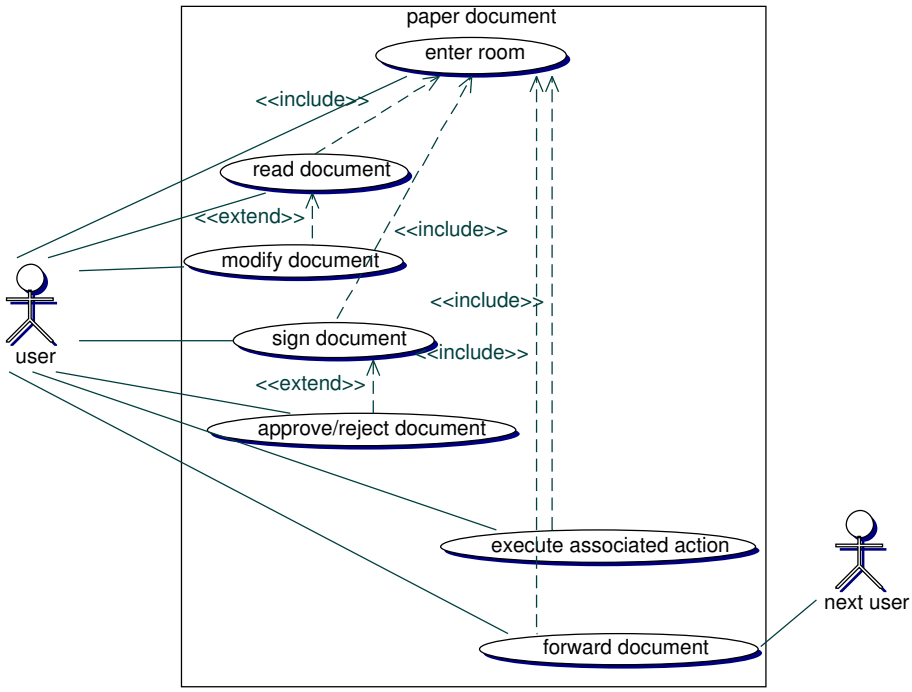


Figure 3.1: This UML diagram illustrates the features of a paper document in form of several use-cases (see also Fig. 3.2).

- The second point is, that a reader can leave his signature on the document or on the folder/envelope which contains it in order to signal and to prove that he has processed the information. Such a signature is also used to authenticate approvals or disapprovals. Of course, signatures can be forged or copied, but techniques and rules to (at least) minimize this risk are well known and established.

- Third, we have to consider that especially in administrative environments, predefined actions are associated with each document respectively upon its receipt. When the document is passed on later, also the next person in the process chain typically knows about the following actions to be performed before it is passed on again or before returning it.

These three features specified above – in computer science they are usually called *access control*, *authentication* and *workflow* – are, besides storage of course, the most important ones that have to be implemented by Document Management Systems or (more general) Information Management Systems in order to replace a traditional paper-based workflow by its electronic counterpart. A transition from traditional work processes to their digital equivalent cannot be achieved with any of them missing. Even more critical, experienced users will recognize problems in these areas well before starting to appreciate the advantages of the new electronic workflow. See Figure 3.1 which displays the features of a paper document and their interrelation as an UML use-case diagram [RJB97]. As a comparison, Figure 3.2 shows how a digital document can implement the same behavior: although the interrelations and the actors have changed somewhat, the general structure as well as the single use-cases remain mostly the same.

Of course, electronic information management systems also include some (hopefully many) of the advantages which an electronic environment can offer. Examples are the simultaneous access to shared documents from distributed locations, the automatic creation of full-text indexes including new documents at insertion time or (full-text) queries and automatization of simple tasks, to name just the most

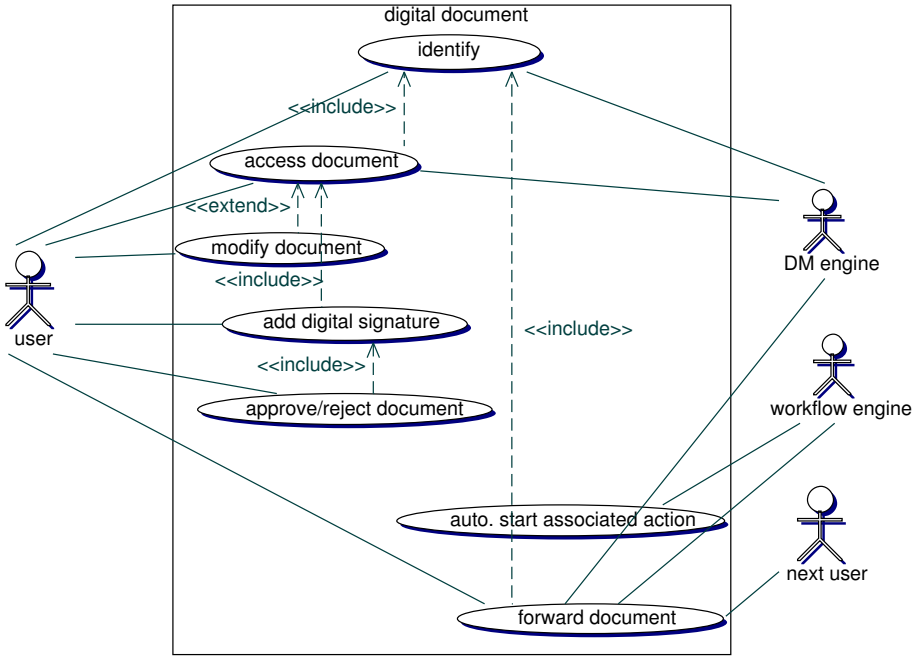


Figure 3.2: This UML use-case diagram shows, how a digital document emulates the features of a paper document (see also Fig. 3.1).

prominent ones (see also in Chapter 2).

Another major point which has to be stressed here is that *any* system implementing the three key features discussed above resembles an Information (or even Workflow) Management System and thus can be used in that way, even if it is not marketed or easily identified as such. We will come back to this point later in this chapter because based on this idea, our group used a 2nd-generation Web server to manage the documents and their workflow in the *GoldenGate* project.

3.2.2 Hyperwave Information Server

As basis of our *GoldenGate* system, i.e. as Information Management and Workflow engine (see in Section 3.2.3), we used a *Hyperwave Information Server* (*HWIS* or simply *HIS*) from *Hyperwave*[3]. Hyperwave (formerly known as *Hyper-G*) is based on research by Maurer, Kappe et al. in the early Nineties [KMS93, Mau96]. This server is often characterized as the first second generation WWW server because instead of simply making available documents in a file-system through HTTP-requests, it is a Document Management System with a smoothly integrated WWW-gateway, offering the complete functionality through an off-the-shelf browser (Figure 3.3 shows an example). Simply speaking, Hyperwave is a combination of an integrated, native database (beginning with Version 4 it became also possible to use well-known existing database systems like those from *Oracle* or *Microsoft SQL-Server*) and a WWW server. It is based on modern concepts of database design and information retrieval as well as multimedia storage and Internet access.

The following is a list of some of Hyperwave's relevant features which we exploited to model our *GoldenGate* system:

- The integrated database (or an external one) stores any kind of generalized document as a (multimedia) data-object. All these data-objects are organized in form of one or several hierarchies

[3] http://www.hyperwave.com

of objects and collections of objects, which in turn are again considered as objects themselves. This complete structure forms an acyclic bidirectional graph. In any case, one data-object is physically stored only once, several instances (at different positions in the hierarchy) are modeled by reference.

Again, these references or bidirectional links are also first-class database objects themselves. Additionally, they are robust, meaning if an object is moved or renamed all links remain nevertheless valid. For example, hyperlinks between HTML documents are automatically represented by such links and thus, dead hyperlinks between internal documents are not possible with Hyperwave. If a document is deleted, all hyperlinks which use this specific document as source *or* destination are also removed automatically (which is possible because the unidirectional HTML links are internally stored as bidirectional references).

- Any object in the database managed by HIS is described by metadata called *attributes*. Some of these object attributes (like `Owner` or `TimeCreated` etc.) are automatically generated and managed by the system, but a user or an application can always add arbitrary additional metadata as necessary. Additionally, all attributes can be indexed by the system if requested. This enables fast boolean or ranked queries, both being supported by Hyperwave.

- HIS offers an integrated user management system, including users and groups which are also represented by database objects (including arbitrary metadata and access rights, see below). Alternatively, external user directories can be used (e.g. with the help of *LDAP*, the standardized *Local Directory Access Protocol* [WHK97]).

- One of the system-controlled object attributes mentioned above is used to model a fine grained access control mechanism, very similar to the access rights of a UNIX file-system. Rights for read-, write- and delete-access can be explicitly specified for single users, for groups of users and for all others.

- The full-text indexing and search engine of *Verity Inc.*[4] is an important part of Hyperwave. It allows full-text queries not only for "plainly readable" documents, but also for popular formats like PDF, PostScript and several formats from the Microsoft Office suite (see Section 2.1) besides many others.

- The access to all Hyperwave features and all stored documents via the browser gateway is based on layout templates. A predefined set is included, but it is easily possible to create own templates and to assign them on a per-object basis. These templates consist of standard HTML-code, a simple macro language called *PLACE* [Hyp01] and last but not least server-side *JavaScript* [KM97].

- HIS offers numerous other features like support for multilinguality, sequences of objects, stored queries for manual or regular repetition at a later time, gateways to other databases and so on...

All these features above (and others which were not listed) can be used and even administrated via conventional WWW browsers anytime from anywhere on the world (given a TCP/IP-connection and proper access rights, of course). Additionally, the full functionality can be used from other programs via an SDK (Source Development Kit) that is available for several programming languages (including *Java* and *C++*). Software based on this SDK can then communicate with the server using either a native TCP/IP-protocol (see also in Section 3.3) or using a newer protocol based on XML-messages embedded in HTTP-requests (which is thus able to tunnel usual firewalls).

As a final point it has to be mentioned, that Hyperwave Information Server runs on several different hardware and software platforms, including such common ones as *Microsoft Windows NT/2000/XP*, *Linux* or *SUN Solaris*.

[4] http://www.verity.com

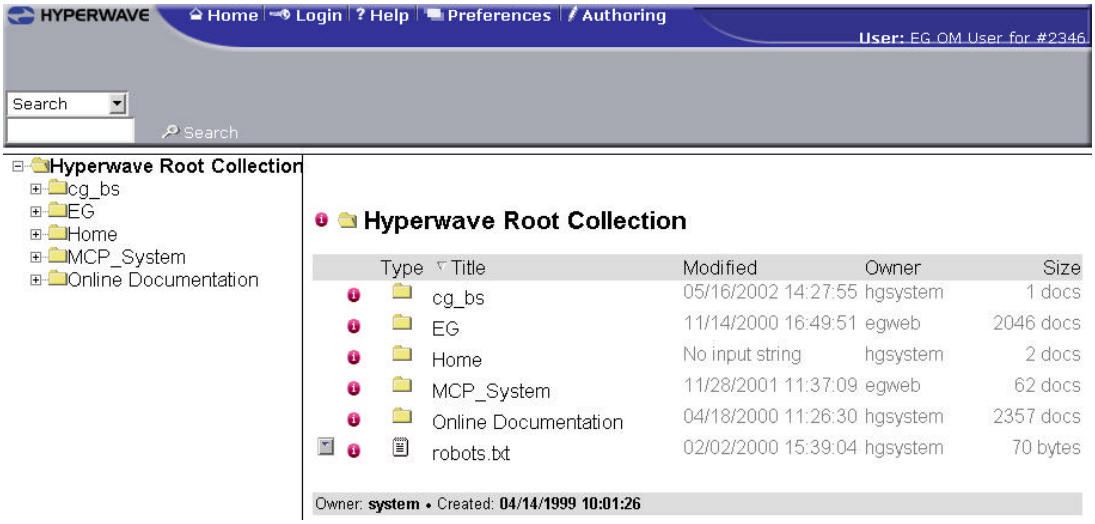


Figure 3.3: A screenshot of *Hyperwave Information Server* displaying a collection with the default layout templates (of version 5.5) in a WWW browser.

3.2.3 Modeling Information Management with Standard Components

Modern Systems for Information or Workflow Management (some are listed in Section 2.3.2, other names are *Lotus Notes Domino*[5] and *InConcert*[6]) do implement the necessary key features as identified in Section 3.2.1, of course. So why follow a different approach, as we did with *GoldenGate*? The short answer in our case is that these software packages often have disadvantages, which are severe in some situations (and may be negligible in others) [FZ99b].

To establish a new full electronic workflow of documents, one traditionally has two options: implement a new, own system from scratch or buy an existing one. Let's have a closer look at both possibilities plus a third one, which we have used to implement *GoldenGate*.

Implementing a New System

Implementing a new, specific (and monolithic) system for a given workflow has one major advantage: it will smoothly integrate into the existing environment and quickly replace and enhance all former paper-based processes (if everybody does his homework, of course). However, this is difficult because a lot of man-power (i.e. a full fledged IT department) and time (usually several months) is needed to build a reliable system from scratch. Both is very expensive, if available at all.

There are several other drawbacks. Even after the new system has been deployed for usage, its maintenance will be very expensive for the complete running time, because the user (i.e. an IT department or similar) has to do it on his own. There will not be any support or helpful experiences from other sides. Another point is the requirement for updates to match the state-of-the-art or to add new interesting features which will also be the users' own task. And last but not least, if the requirements of the application change over time, there will be a potential demand for reengineering or even a complete rewrite in order to adapt.

All in all, building an own system might be a viable solution for a short duration and a limited range of applications, but in the long-term the situation might become very complex and expensive. So what about the second option mentioned?

[5] http://www.lotus.com
[6] http://www.tibco.com/solutions/products/active_enterprise/in_concert/default.js

Deploying an Existing System

Buying and deploying an existing Workflow and Information Management Solution has some advantages when compared to the previous option of implementing a new system. Of course, this results in the requirement of much less software development on the users' side or even none at all. Additionally, the availability of the external basic package and of future extensions is guaranteed. Maintenance and upgrades would be provided from a third specialized party for a long time.

But, as always, there is another point of view. For example, it wouldn't be the first time that a company and its product vanish from the market. The users have no influence at all and would be left alone to organize maintenance and upgrades of their "old" system. Even if the company remains, at some point in the future they might say that they won't support the system anymore and that the users should buy a new one. A more important point surely is, that if someone buys a common tool not being tailor-made for his problem, he might have to live with a comprise. This means that the user would have to adapt his office and his workflow processes instead of the tool. Many companies and organizations can tell a lot of stories about this (e.g. when deploying a de-facto standard software from *SAP*[7]). A major problem of such a situation is that the actual users have to be re-trained, which is again expensive as well as time-consuming and might lead to an "unhealthy climate" within the staff.

So, there are advantages against the previous option and most often this approach will be the right one to choose, but also here there are several problems. These consequences encouraged us to look for a third and more appropriate alternative for our planned *GoldenGate* system. See below.

Combining Standard Components

Our new, third approach to implement an Information and Workflow Management System during our research project *GoldenGate* is based on the idea of combining independent software components to build a new, more complex system [FZ99b]. One has to admit that especially in Computer Science and its field of Software Engineering this idea is not really innovative, but when we decided to follow this road in 1997 practical experiences where generally missing. What is new though, is that we used a special kind of basic components: standard tools, meaning office software and Internet technologies which are already in use.

When building the new system from components that the user already has *and* uses, he or she gets the following advantages nearly for free:

- The new system will smoothly integrate into the existing environment. Remember, the core components are already part of this environment and the remaining "glue" is tailor-made to fit.
- The user will get a predictable and guaranteed functionality (because he already has it, otherwise he wouldn't use the combined tools).
- The deployment of the new system will be very efficient because there is less (read: almost none) program code to write or to maintain and there is no need to re-train the users.
- Last but not least, whenever one of the embedded components improves (i.e. an update by the manufacturer offering new features) the whole system also improves automatically.

As a conclusion, the basic idea behind the new architecture which we developed for our *GoldenGate* Information and Workflow Management system (i.e. the "house" in Figure 3.4) is to use standard office and Internet tools as components (i.e. the "bricks") and to combine them with a thin layer (the "cement") which is especially implemented and tailor-made for the specific application.

[7] http://www.sap.com

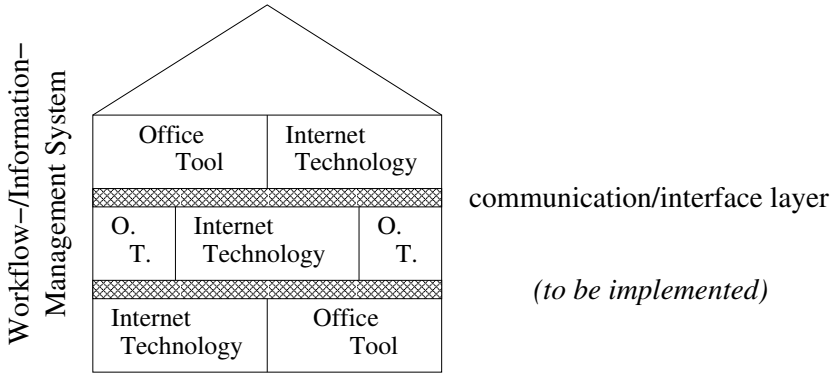


Figure 3.4: An Information Management System formed of office tools (O.T.) and Internet technology (the "bricks") combined by a thin communication or interface layer (the "cement").

3.2.4 Bringing it all Together: *GoldenGate*

Each current Information Management System consists of several tightly integrated functions or interfaces, which can be grouped according input, output, access, storage and workflow. If these components are integrated into one complete system, everything works smoothly and efficiently together (see also the previous section). However, problems arise when applications require the use of special input components which are not included in the integrated system. Another problem often is individual worldwide access (for collaboration and communication, often called *Groupware*), especially using the complete functionality without special (and expensive) client software, if possible.

Many research groups are working in these fields, stressing one or the other aspect. For example, the *Open Hypermedia Systems Working Group*[8] tries to combine existing tools as components and to extend them with Hypermedia functionality. The *Basic Support for Cooperative Work* project (*BSCW*) at GMD (now a Fraunhofer Institute) [BHST95] concentrates on cross-platform group collaboration services based on existing WWW technologies. A third example is the also Web-based *Endeavors* [HBT97] from U.C. Irvine, which is a flexible and lightweight workflow infrastructure that provides support for process specification, distribution and integration of third party tools.

Our approach for *GoldenGate*, which is based on the new idea from the previous section and whose implementation is given in detail in Section 3.3 below, can be characterized best by: take the software which is already used for input, output and handling of documents plus a *Hyperwave* server (see Section 3.2.2) for the storage and workflow as the basic components and combine these using the Internet technology.

As a consequence of this way to build the system, the users at DFG can continue to use their favorite office tools (i.e. *Microsoft Excel* and *Word*) to enter information and to store it in the applications' native formats into the *Hyperwave* database (which can handle them as any other type of data), provided proper access rights have been granted. As the information is stored in the native format of both tools, *Word* and *Excel* can again be used to extract, to modify or to output the documents. All this is independent from the physical location of the user (given a network access is available), because the full functionality of *Hyperwave* is accessible through a WWW browser and other ordinary Internet technologies.

Storage and access control is one of the basic features of an *Hyperwave Information Server*. For example, personalized users hold an account with username and password, in contrast to anonymous ones. Together with the fine grain access control at the object level this allows a detailed tuning of document access for reading, writing, modifying or following a hyperlink, with anonymous users having access to public information only.

[8] http://www.ohswg.org

It is important to note, that, due to being fully programmable, common office tools are very flexible today (e.g. see Section 3.3). As a consequence, customization to individual demands with regard to user functionality or user settings can easily be accomplished. Such customizations will in turn enable the various components not only to download or upload data but also to interact with the *Hyperwave* server. Components can store/modify metadata about documents, query for additional information or collect data from a hierarchy to build a new document (e.g. a statistical analysis within *Microsoft Excel*, see below).

The latter feature of the used components is the key to the implementation of the final part of an information management package: the workflow. The basis for workflow management are processes and use-cases. Someone enters a new document and opens a case for a specific process. A second person modifies the document, a third one returns it to the second or perhaps closes the case. Such a process can easily be modeled in a document oriented fashion by passing along proper access rights. While the first user creates the document this person is the only one with full access. When the document is passed on ("checked out"), a tool adapts the access rights such that only the second person can work with and modify the data and so on. *Hyperwave's* built-in features like collections and reference linking can be used to have such tasks (i.e. documents to work with) show up in special (i.e. personal) locations of the database hierarchy, with the recipient being informed about the arrival either by email or by the result of a stored query (another *Hyperwave* feature).

All in all, this sounds like a lot of work. In fact, there is work to do, but significantly less compared to building a full system from scratch. According to our approach we only need to build the interfaces between the components which are already in use, similar to inserting an integrated system into an existing environment. Considering that we build on top of office components which the users at DFG are already familiar with, the training with the new system can be kept to an absolute minimum and productivity will not be reduced much during an adaption period.

3.3 Implementation Details

Before several interesting details of our implementation of the *GoldenGate* system will be discussed later in this section, an overview about its components and the general workflow will be given.

Following the ideas of the previous section and taking into consideration the requirements within DFG (see Section 3.1), we used the following design:

- a *Hyperwave Information Server* acts as a central (though possibly distributed) database for digital multimedia documents and access point including user management,
- *MS Office* applications (i.e. *MS Word* and *MS Excel*) as well as
- ordinary WWW browsers (e.g. *MS Internet Explorer* or *Netscape Communicator*) play the role of access and administration tools.

The typical workflow starts with possible applicants downloading (via their own Web browser) an application template for their favorite word-processor (namely LaTeX, *MS Word* and *Adobe FrameMaker* are supported). For the researcher requesting funding this template contains support and help (by examples, "wizards" and selection lists of possible values) when creating the application. The resulting digital document is uploaded (again, via a Web browser) to the *GoldenGate* server located at DFG, where it is converted into our internal format (i.e. XML) and stored in a collection at a special node of the database hierarchy. See the Sections 3.3.1 and 3.3.3 as well as Figure 3.5 for a further discussion.

Upon arrival of a new grant application form at the DFG's server, an officer in charge is informed (either by email or by a look at his personal "new applications" collection). From this application a so-called *dynamic view* (i.e. a *MS Excel* sheet) is generated automatically and also stored in the database.

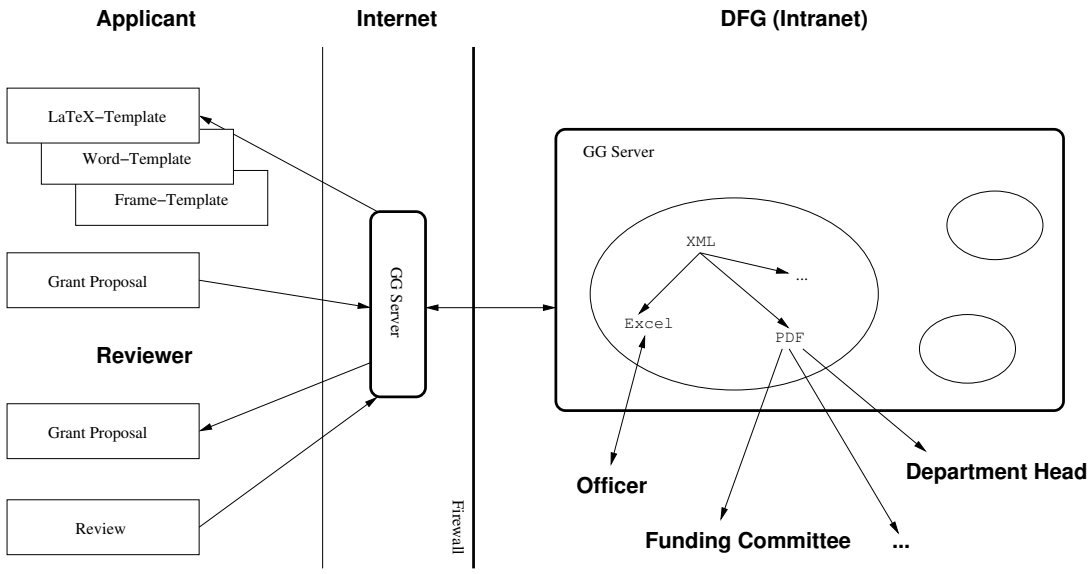


Figure 3.5: The typical workflow for a research grant proposal when processed by the *GoldenGate* system.

It is referenced from other locations within the hierarchy, e.g. according to one or more area-specific thesauri, and it is supplemented by metadata which was automatically extracted from the application form during conversion.

Those dynamic views can then be checked, edited or passed on with *Excel*. Many features of this program like pop-up menus, forms or *COM* [Ise00] support the user at DFG in the subsequent handling of the grant proposal. For example, he or she can access the internal address database (consistently maintained within the same *Hyperwave* server) or choose standard salary amounts for researchers from various central lists. During the whole process so-called *static views* (actually we used PDF documents, but pure text or anything else is possible) can be (and are by certain tasks) generated at any time to freeze a special state or to act as read-only documents for other departments or external reviewers. See also in Section 3.3.2 below.

All the time, the documents remain stored in the database hierarchy of the *GoldenGate* server. "Free" browsing or non-standard queries to the database, i.e. access to the DB which is not yet part of a dedicated module directly accessible through the user interface, can be done with each user's favorite Web browser using the default front-end of *Hyperwave*. For example looking at all proposals by applicant "X" or at all accepted applications in research field "Y" is possible – given the proper access rights, of course. In case someone clicks on a dynamic view to work with it, the document is checked-out by assigning the read-write access to this user exclusively until it is checked-in again. The programmable layout templates of *Hyperwave*, its internal locking mechanisms and CGI scripts are used to implement this behavior. See also Figure 3.5.

During first demonstrations and iterations we learned quickly that we had to increase our system's abilities as compared to the first design, which we did successfully: support for standard letters (like for acknowledging the receipt of the proposal, asking for missing details...) and mail merges, storage for internal and external documents (like letters to the applicant or the reviewers) and statistical analyses of elements in the database became necessary. Particularly in this situation our ideas did prove their power and their flexibility, because we had not to reinvent anything. We just integrated more of the existing features of those tools we already used and only had to update some interfaces and front-ends. Now *MS Word* is used to generate mail merges by accessing the address database on the *GoldenGate* server and

1. Allgemeine Angaben
   1.1 Antragsteller, 1.2 Thema, 1.3 Kennwort, 1.4 Fachgebiet und Arbeitsausrichtung,
   1.5 Voraussichtliche Gesamtdauer, 1.6 Antragszeitraum, ...
2. Stand der Forschung
   2.1 Stand der Forschung, 2.2 Eigene Vorarbeiten/Arbeitsbericht
3. Ziele und Arbeitsprogramm
   3.1 Ziele, 3.2 Arbeitsprogramm
4. Beantragte Mittel
   4.1 Personalbedarf, 4.2 Wissenschaftliche Geräte, 4.3 Verbrauchsmaterial,
   4.4 Reisen, 4.5 Sonstige Kosten
5. Voraussetzungen für die Durchführung des Vorhabens
   5.1 Zusammensetzung der Arbeitsgruppe, 5.2 Zusammenarbeit mit anderen Wissenschaftlern,
   5.3 Auslandsbezug, 5.4 Apparative Ausstattung, 5.5 Laufende Mittel für Sachausgaben, ...
6. Erklärungen
7. Unterschriften

Table 3.1: This table presents an overview of the required document structure of a grant proposal to DFG.

to store arbitrary digital documents. The statistical components of *MS Excel* (like tables, bar graphs, pie charts...) are used, grabbing the necessary data simply from the attributes of the database hierarchies.

3.3.1 Submitting a Grant Proposal

The critical factor for the success of any digital document workflow is the acceptance of the end-user (in this special case: researchers submitting grant proposals). The new workflow has to be at least as comfortable as the previous paper-based one and it has to offer a clear advantage. In the pre-digital age (which still seems to last at DFG) applicants had to provide their grant proposals as a printed document, its structure and content following some strict guidelines which were defined in a brochure of about one hundred pages size. Have a look at Table 3.1 for an overview of the required structure. Of course, almost nobody reads such documents in its entirety. Instead most people try to find an example or a previous application and derive their new document from the older one. For our *GoldenGate* system we decided to also follow this road. So we had to provide digital example documents in a format that all possible applicants could and would use without problems – it is obvious that the latter is very important for the acceptance of the whole system.

Therefore we prepared three application templates as *MS Word*-, *Adobe FrameMaker*- and LaTeX-documents. Those documents exactly followed the structure and content guidelines of DFG for grant proposals and thus allowed all end-users to adhere to the correct form for paper-based submissions when printed (see Figure 3.6). But we made also the next step, i.e. to offer an advantage. We used the scripting abilities and GUI elements of *MS Word* to enhance the template for this word processor: it became an active document.

We added dialogs and programmed forms (today often called "wizards"), which on one hand help the author to fill the right document sections with the correct content (e.g. he can select the possible funding for research staff from a listbox, see Figure 3.7). On the other hand these functions allowed to check for structural and logical correctness, giving the author an immediate feedback instead of having to wait for a call from the DFG officer in charge. Other advantages of this approach include the possibility to update a document at several places automatically if one value effecting many positions is changed (e.g. the authors name on the cover sheet and below the signature at the end).

An important feature of our document templates (not directly for the author but for DFG) is the hidden markup, i.e. invisible "hints" or "tags" embedded into the document. This metadata makes it

Antrag auf Förderung eines Forschungsvorhabens durch die DFG

1 Allgemeine Angaben

1.0 Art des Antrags

Fortsetzungsantrag

1.1 Antragsteller

Prof. Dr.. M. Mustermann, Universitätsprofessor
Geburtsdatum: 11.1.1911, Nationalität: Deutsch
Geschäftszeichen des Vorantrags: Mu-17/2

Figure 3.6: The "print-like" view of a grant proposal created in *MS Word* using our template.

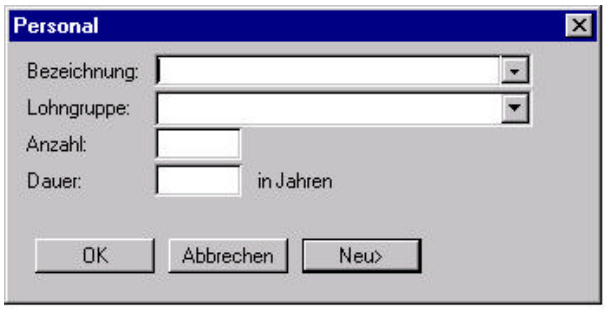


Figure 3.7: This is an example of the GUI dialogs available in the *MS Word* template. The data entered is inserted into the document at the right position (or positions) using the correct formatting.

possible to safely and reliably extract important data automatically when the proposal is finally submitted to the *GoldenGate* system. See Section 3.3.3 below for a thorough discussion of this mechanism. In the *MS Word* version of the application template this markup is implemented with paragraph and character formatting styles. For example, when the applicant's name is entered, it is marked with style `ggName`. After the document is stored as *RTF* (which is our transport format for *MS Word* documents, see Section 2.1.3), the resulting file is parsed by the system, relevant information is extracted and an XML file is generated which will include the text with this style as content of the element `<ggName>`. The presentation-oriented *Word* document is automatically converted to structured information (see Section 2.1.2) with the help of the embedded markup. We can then use one of several available XML parsers to check the new application and (possibly automatically) reject it if something is missing or the structure is incorrect. The necessity to keep this special formatting styles intact in order to be able to automatically convert the document later on, is another reason why we implemented GUI elements like dialogs and forms: under *Word* it is very easy to unintentionally change or even delete the style of a paragraph or a character when manually editing this part of a document (see also Sections 3.3.3 and 3.4.2). The risk of doing so is minimized, when the author uses a form and the document itself is updated by our own *VBA* (i.e. *Visual Basic for Applications* [Zak00], the scripting language of *MS Word*) macros.

The author can download the templates described above from a DFG Web page (e.g. from the system's *Hyperwave* server), fill them out, modify them or print them (which will result in a well-layouted paper document, see Figure 3.6) until he is satisfied with his grant proposal. Finally, the electronic document will be saved (the *Word* document as *RTF*, the *FrameMaker* version as `.mif` and the LaTeX one as `.tex`) and can then be submitted to DFG (i.e. the *GoldenGate* system) through a WWW form. On the server side, a so-called *dynamic view* (see in Section 3.3.2 below) is created automatically which allows the officers in a DFG department to fulfill all typical tasks (see also Figure 3.5 for an overview of the whole process). In the meantime, the researcher is informed about his successful submission by email.

The upload of a grant proposal to the *GoldenGate* server as described above requires a clear identification of the researcher responsible. To make this possible, he has to register himself as a first step. This is done by simply entering name, address, email address and the DFG internal identification code ("DFG Stammaktenzeichen") (if available) into a WWW form (see Fig. 3.8) and submitting it. The *GoldenGate* server can then check (or update respectively) its internal address database and possibly generate a new identification code. Additionally, the server will create a unique user account and password which is then sent to the email address specified. The researcher must use this account to upload his grant proposal and is thus identified.

Last but not least, the purely digital workflow offers another advantage for authors of grant proposals: it is now possible to supply additional multimedia documents as attachment to the application. For example, the researcher can provide video, audio or additional text material by simply uploading it through the same web form as the original document. This material can then be easily accessed by the DFG officers and later on it is automatically forwarded to the reviewers of the application. Of course, a similar mechanism would have been possible with the non-electronic workflow. The author could possibly have provided a video tape or an audio CD-ROM. As a matter of fact, DFG had to reject this, because they did not have the capacity to duplicate these media and to additionally forward them to the reviewers by mail. Set aside whether the reviewers would have accepted and made use of this material...

3.3.2 Dynamic & Static Views

When the grant proposal has been submitted by its author in digital form, the workflow has just begun. From now on, the digital document remains stored within the central database (i.e. a *Hyperwave Information Server*, see Section 3.2.2) of the *GoldenGate* system. It is accessed and modified by ordinary WWW browsers and by the *MS Office* application *Excel* in form of so-called *dynamic* and *static views* (see below).

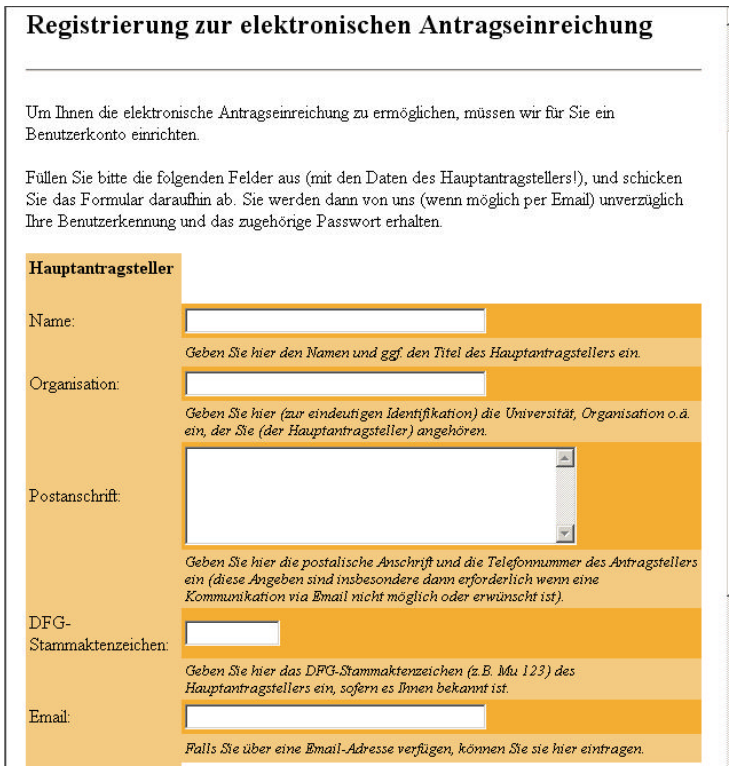


Figure 3.8: Using this WWW form an author of a grant proposal can register for a *GoldenGate* system user account, which is then used to upload the document.

The main components of the system are combined with and accompanied by several tools and interfaces programmed in the scripting language *Python* [RJ99]. Additionally, the Internet protocols HTTP and TCP/IP as well as *COM* (i.e. the *Component Object Model*, Microsoft's software component architecture for *Windows* operating systems [Ise00]) are used for interaction and data transfer. Figure 3.9 gives an overview about the structure of our prototype implementation.

As can be seen, all components are centered around a *Hyperwave* server holding the *GoldenGate* database, which follows our paradigm to combine standard components to build a new, complex system (see Section 3.2.3). An important part of the database's hierarchical organization is schematically displayed in Figure 3.10. The main sections are the `Personal Homes` and the `Applications` collections. The former contains one sub-collection for every "user" (be it a researcher having submitted a grant proposal or a reviewer or both, which is often the case) and the latter consists of sub-collections for all applications which are currently active or have been active a limited amount of time ago. Both collections are initialized with an address database and all documents relevant for currently active research grants. These are extracted from the conventional DFG system (*All-in-One*, see at the beginning of this chapter), parsed and inserted by *Python* scripts, when *GoldenGate* is set up. Exploiting the features of *Hyperwave*, other virtual hierarchies (e.g. an alphabetical hierarchy of persons, a hierarchy for several research programs or a theme thesaurus to quickly access special fields of research) are added according to the demands of the individual DFG departments.

The supplied *Python* scripts (like the one to initialize the database structures) communicate with the central *Hyperwave* server using the native, TCP/IP-based Client-Server protocol (*HG-CSP*). Following the concept of modularization, we implemented this task by creating an API for the *HG-CSP* as a C++-extension to *Python*, which is embedded via dynamic runtime linking (i.e. a *DLL*, as *MS Windows NT* is the computer platform). This API is also used by other *Python* tools like the address dialog (see Figure

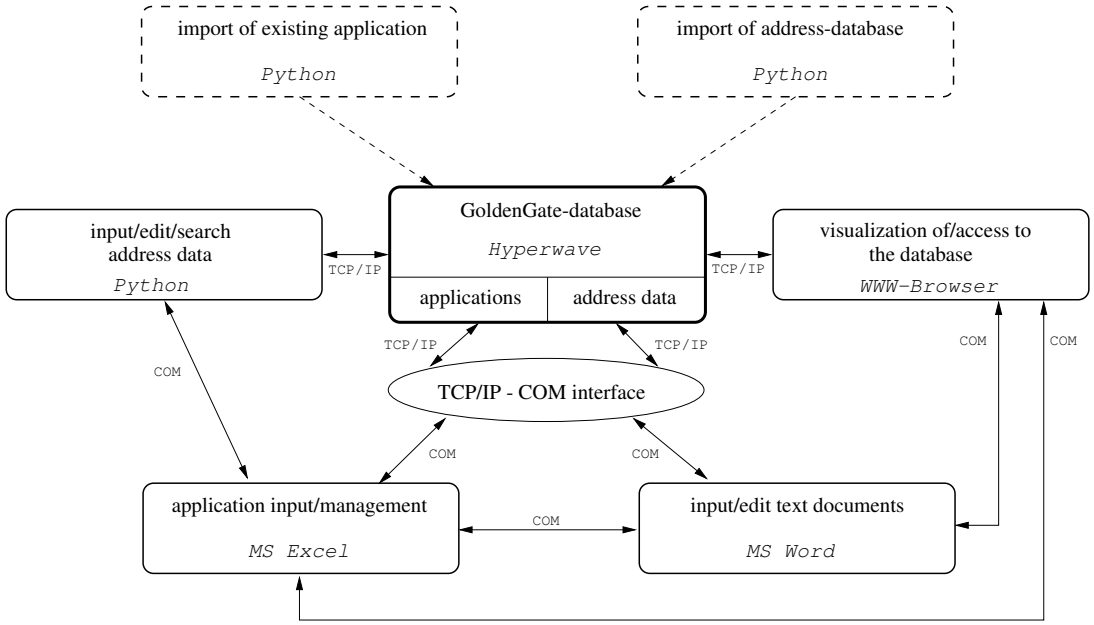


Figure 3.9: The structure of the *GoldenGate* system as implemented by our prototype.

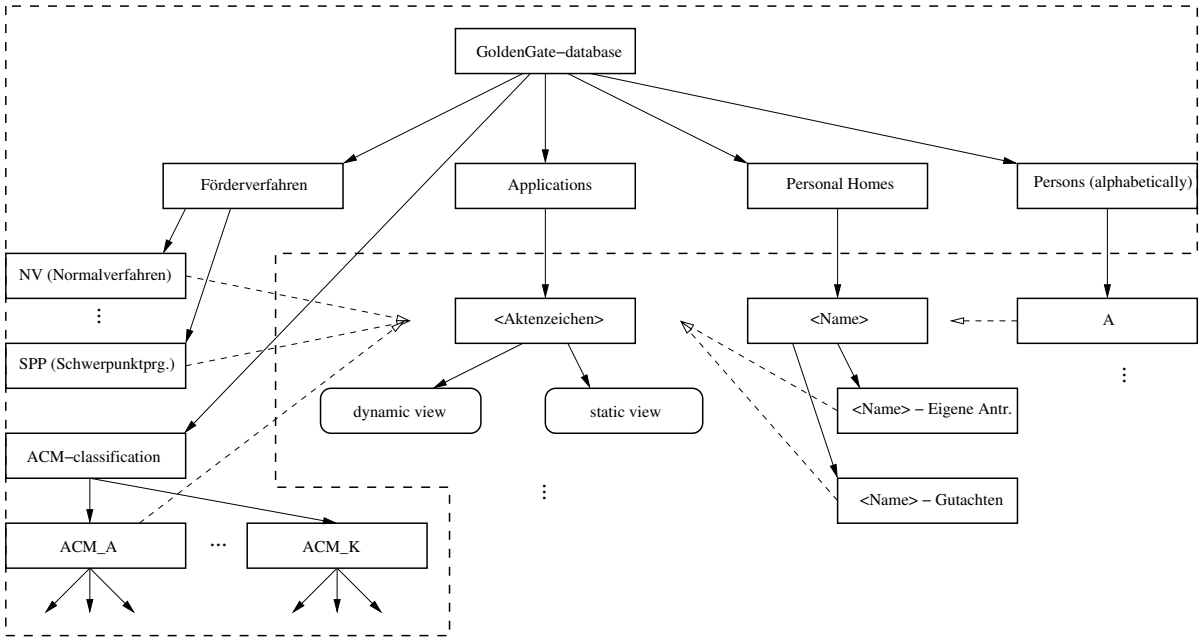


Figure 3.10: The hierarchical structure of the *GoldenGate* database. The part within the dashed border is generated automatically, dashed arrows show references.

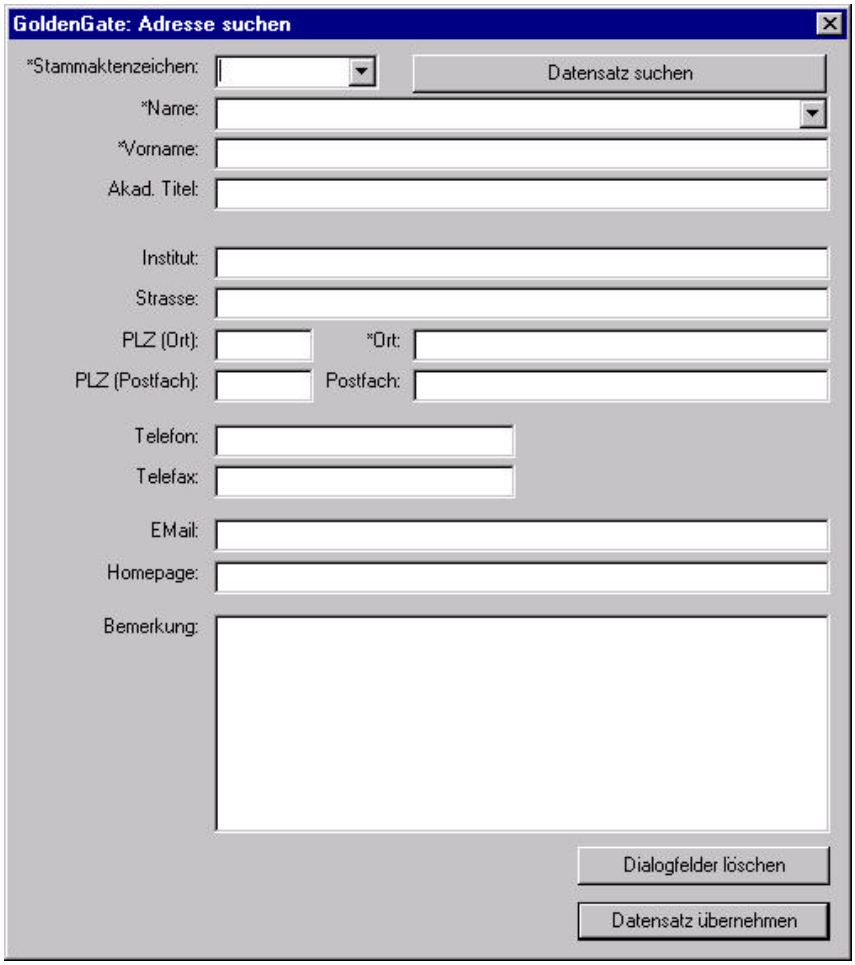


Figure 3.11: The address dialog, one of the *Python* tools in the *GoldenGate* system.

3.11) which acts as a graphical front-end to search, edit or update the address data. Other examples are several hierarchy browsers being used by *MS Excel* and *MS Word* to directly access objects on the server. The latter tools are implemented with the help of Mark Hammond's Win32-extensions to Python [HR00], which amongst other features allow to use *Windows* GUI elements and system calls.

Another important role in our system plays the so-called *TCP/IP-COM interface* (see also in Figure 3.9). The Win32-extensions to *Python* allow a script in this language to act as a *COM* client, i.e. a software component which uses functionality (including methods and data objects) that other components (e.g. system libraries, processes or applications, being called *COM* servers) offer via the *COM* protocol. One *Python COM* server is registered for the address dialog. As a result *MS Excel* (which like all office applications from *Microsoft* is a *COM* client, of course) can use it easily and transparently. Several other servers form the already mentioned *TCP/IP-COM interface*. This interface offers methods, objects and GUI dialogs to access *Hyperwave*, so one could say that *COM* messages are translated into calls of the *HG-CSP*. This setting allows the office components *MS Word* and *MS Excel* to access, load, store and even organize documents and collections in the server's database.

A visualization of database hierarchies (see Figure 3.12), objects and metadata (like the one automatically extracted from submitted grant proposals, see above) is presented by *Hyperwave* using HTML. Thus, one can simply use any WWW browser to access documents and information. We modified the server's layout templates (using a combination of *JavaScript* [KM97] and *PLACE* [Hyp01], the native

Figure 3.12: The structure of the *GoldenGate* database hierarchies is visualized by *Hyperwave*.

template language of *HWIS*) which are responsible for visualization to suit our needs. For an example have a look on the Web page displayed in Figure 3.13. Here one can see how metadata is extracted from database objects and presented, when the user looks at the contents of a collection containing a grant proposal. However, the major role to work with these proposals within DFG is played by each application document's *dynamic view*, which can be opened from the collection (see button `Dokument 'Antragsspiegel (dynamisch)' bearbeiten` in Figure 3.13).

What we call *dynamic view* is a special spreadsheet document for *MS Excel* which is generated and filled out automatically, when a DFG officer in charge accesses a newly submitted grant proposal for the first time. The template for this spreadsheet (a `.dot`-file) – besides including an empty default sheet and structures – activates buttons and menu entries which can initiate the execution of several *Visual Basic for Applications* (*VBA*) (e.g. [Zak00]) modules. See Figure 3.14 for an yet empty document. Upon execution of the "import" module (by simply clicking on one of the buttons in the *GoldenGate*-toolbar, added to *Excel's* user-interface), the user at DFG only has to decide which new grant proposal to read, as several might have stacked up, e.g. over night, by choosing it from a list dialog (i.e. one of the *Python* tools being accessed through the *TCP/IP-COM interface*, see above). The according XML document (see below in Section 3.3.3) is parsed and the newly created sheet is filled out automatically. The officer only needs to assign a new grant ID (i.e. "Förderkennzeichen", used by DFG as an internal unambiguous reference), check the plausibility and store this new *dynamic view*, as we call it, into the database. Figure 3.15 shows an example of such a filled-out document.

As an additional automatic step during the filling out of the *Excel* sheet, the "storage" *VBA* module builds a collection for the whole new process (see Fig. 3.13), references it from all applicable places (see the dashed arrows in Figure 3.10) and creates a so-called *static view* in PDF (by simply "printing" the spreadsheet through *Adobe's PDFWriter*). This readily layouted and read-only document acts as a snapshot of the application's current state. Such snapshots are automatically created at several points during the workflow (or anytime on manual request) and never deleted. They allow a tracking of possible changes (of the application as well as of the *dynamic view*) and can be read by several people (with appropriate access rights, e.g. the head of a DFG department), as opposed to the *dynamic view* which will be locked (see below) when being edited.

The *Excel* spreadsheet document was developed in close contact with our partners at DFG. It is

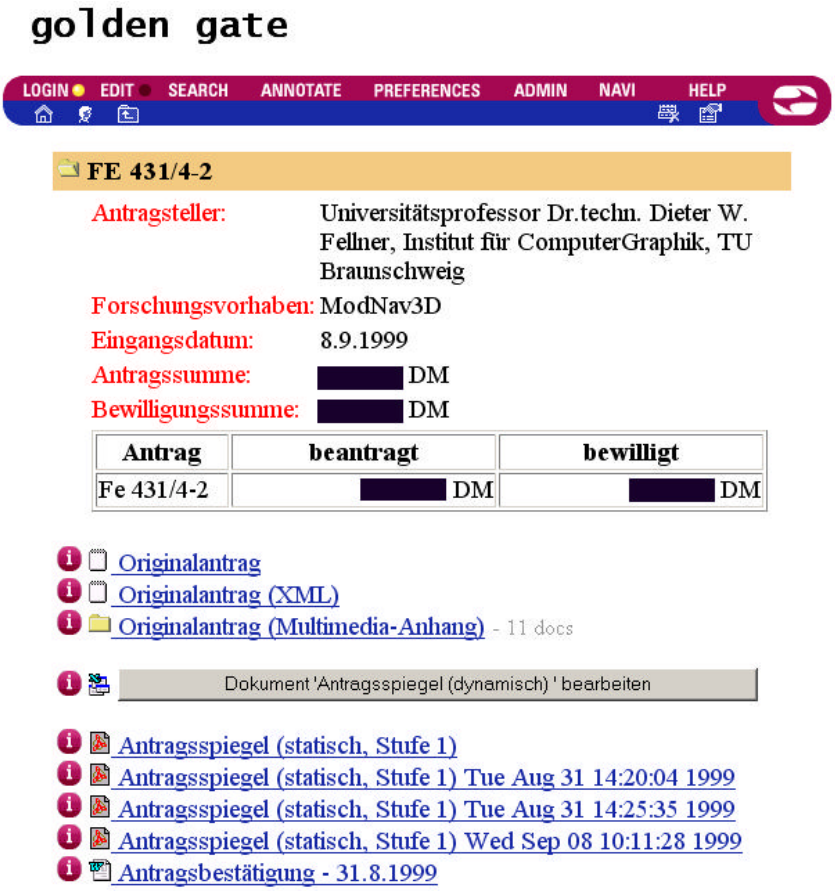


Figure 3.13: Part of the metadata automatically extracted from the grant proposal is presented by the *GoldenGate* layout templates as an overview.

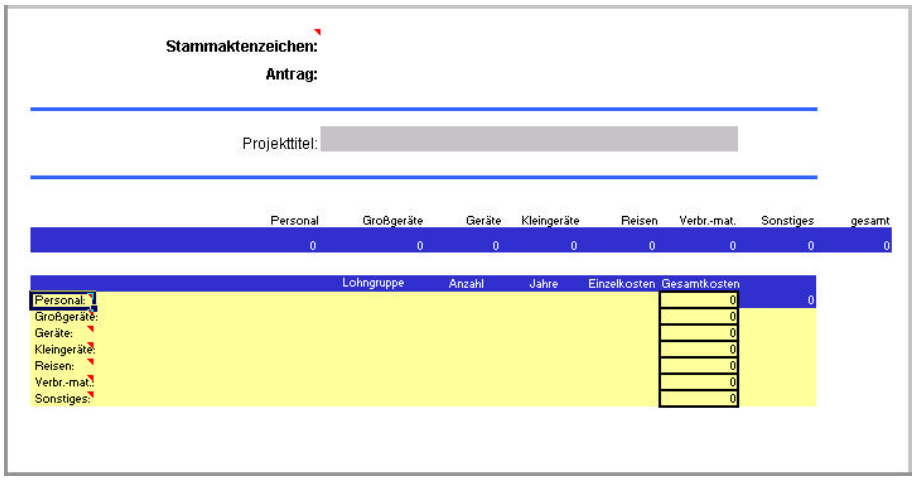


Figure 3.14: A new *dynamic view*, the empty *Excel* spreadsheet as created by the *GoldenGate* template.

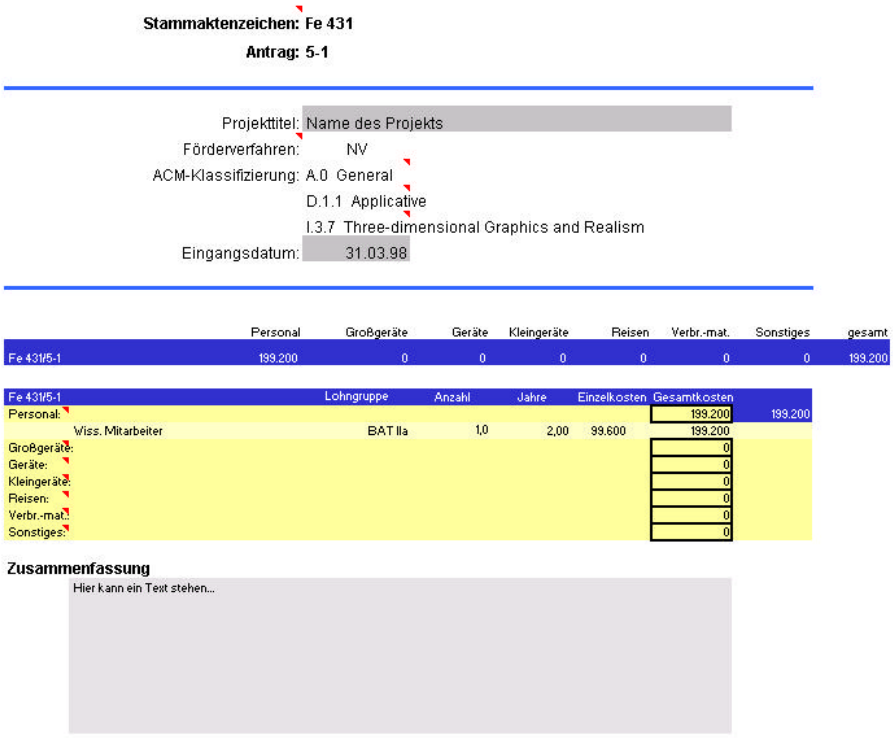


Figure 3.15: This *dynamic view* has been automatically filled with the values extracted from a grant proposal.

the one which is consistently used during the lifetime of an application for research funding, from the submission until the approval (or rejection). The *dynamic view* contains the metadata the most important to the DFG officers, i.e. besides the grant ID and the applicant's name, the money values of the funding requested, split in sections for personnel, equipment and travel funding etc. (as in the required structure of the grant proposal itself, see Table 3.1). These values are often changed based on current rules or after discussions with the applicant. Finally, after the acceptance of the proposal, they are automatically copied into another column for the granted funding. So both, the requested and the granted amount of money can be looked at in comparison. So, as a conclusion, this digital document makes the daily routine work of DFG officers more easy and more efficient (see also Section 3.4 for enhancements which were added later, based on more experiences).

Other important *VBA* modules embedded in the spreadsheet document access the address database integrated into the *GoldenGate* system (see Fig. 3.11), offer support while editing, implement a simple document history and write back changed values to the metadata attributes of the collection. But especially, they care for the locking of *dynamic views*. The locking mechanism implemented in the prototype system acts as a proof of concept how revision control and workflow management can be implemented easily with the features of *Hyperwave*, i.e. with the shifting of access rights and object write-locks. While one user works with a *dynamic view*, it is locked in the database and an attribute marks it as "in use by X". This is also visible in the WWW browser, of course. It is implemented by a CGI-script on the server and by a *VBA* module at the client, as it is possible to open such a document from both *MS Excel* or the browser. Using this mechanism, we effectively prevent two people from editing the same document, which would otherwise be possible as technically it is downloaded from the server and uploaded later on. Additionally we provide a display of who is in charge of a specific grant proposal at any moment. Remember, read-only access is continuously possible by using the latest *static view*.

As presented in the previous paragraphs, much data is not only stored in the grant proposals or in the *dynamic views* but also in attributes of database objects. This allows to access and to reuse it easily or even to have it indexed by the database for faster queries. For example, this metadata is used to display dynamically generated (i.e. always up to date) overviews in the Web browser (as hinted at in Figures 3.12 and 3.13). As another example, this information can be collected within a complete branch of the database hierarchy. In this way it is possible to always see (again in any ordinary Web browser, anytime, anywhere) the current numbers about funding in a special program, in a specific research field or in all cooperations with another country, only depending on the collection structure and on the database queries which are formulated. The third and most important example were these numbers are used successfully is during the final committee meeting which decides about all grant proposals for a special research programme. For this task we developed another *Excel* sheet, similar to the *dynamic view*, which contains only some important values but for all proposals in question. The cells of this sheet are filled from the available metadata automatically online and it can then be used and modified in offline discussions (e.g. increasing the numbers for some grant proposals and reducing them for others). Finally, the updated values are automatically written back into the database, i.e. into the *dynamic views* and into the metadata attributes, replacing are formerly time-consuming and error-prone work.

3.3.3 XML Up-Translation

As already mentioned at the beginning of this chapter (see Sections 3.2 and especially 3.2.3) we wanted to receive the documents containing grant proposals in digital form. For a complete electronic workflow it is essential that the data important for later steps of the process which is contained in these digital documents is inserted in the *GoldenGate* system automatically (instead of entering it manually, as it is the case with the original system in use at DFG). If such an automatic procedure is in use, then even first consistency checks or statistical analyses can be run already at this time. During the development of our system we came up with a flexible, efficient and modern solution to this task [Obe99].

At a very early stage of the project we made the decision that we wanted to use XML as our internal document format, because it is structure oriented, it is flexible, platform independent and widely supported (see Section 2.1.5). On the other hand, XML is not the format of choice for the end-user, at least not before modern and wide-spread word processing systems use it (which was definitely not the case when the research project was started in 1997). Instead we wanted the end-users (i.e. researchers submitting grant proposals) to continue to work with the applications they are used to, like *MS Word*, LaTeX or *Adobe FrameMaker* (see also in Section 3.2.3).

The problem is that the usual word processor document more or less (more in the case of *MS Word*, less for LaTeX) contains only presentation markup, but we needed structural markup (containing single, semantically identified data elements, see Section 2.1.2). Therefore we needed a conversion step from the specialized, proprietary and presentation-oriented markup (i.e. a low level of abstraction) which we receive from the applicants to a generic, semantic and structure oriented markup (i.e. a high level of abstraction) that we want to use. This process is called *Up-Translation*, or *XML Up-Translation* when this markup language is used for both the source and destination format (see also [Obe99]).

Generally, there are four possible options on how such a conversion can be achieved. All have their advantages and disadvantages:

explicit pre-tagging Here the author simply adds the correct markup around crucial data, as he should know best. The extraction (or *data-mining*) problem would be solved immediately, even a simpler tagging than XML could be used. But, the author would be burdened by additional, error-prone work and the resulting document would look different (human readers are often bothered by "unnecessary" words and symbols).

forms Offline or online forms are often used, especially for detailed data. They enable a clear identification of data fields (and as such are similar to the explicit pre-tagging above) and even allow some

```
[...]

<!ELEMENT antrag (allg_ang?, forstand?, ziele_ap?, b_mittel?, vorauss?,
                  erklaerg?, untersch?, anl_verz?)>

<!-- ==================== -->
<!-- 1 Allgemeine Angaben -->
<!-- ==================== -->

<!ELEMENT allg_ang (ant_art, antstell, thema, kennw, fachg, v_dauer,
                    ant_zeit, beginn, zusfassg)>

<!-- -->
<!-- 1.0 Antragsart (Neuantrag oder Fortsetzungsantrag) -->
<!-- -->

<!ELEMENT ant_art (neuant | fortsant)>

<!ELEMENT neuant EMPTY>
<!ELEMENT fortsant EMPTY>

[...]
```

Figure 3.16: A small section of our DTD for the internal XML-documents (i.e. the converted grant proposals). Compare to Table 3.1.

form of on-the-fly checking. But online forms are impractical and offline forms need special client software. Both are not well suited for full-text, complete documents or multimedia.

text/pattern matching These are useful to identify fixed strings (e.g. check for their existence as a "switch"), interesting special data types (e.g. a price or a date) or even structural elements based on predefined headings. But these methods require all necessary strings to be identified and support only a manageable amount of variations (e.g. there are *many* different ways to write a date).

analyzing internal markup This option is based on the internal structure of a digital document. Here the layout should correspond to the required document structure (possibly in an invisible way, i.e. different formatting with the same look) which can be achieved by special document templates. But all document models of usual word processors are different from each other, so a conversion could be different and sometimes non-deterministic.

Based on the pros and cons mentioned as well as on our own requirements, we decided to implement the last method. We came up with the following strategy for the *Up-Translation* within *GoldenGate*: our source documents for data are word processor documents in the formats DVI (LaTeX), RTF (*MS Word*) and MIF (*Adobe FrameMaker*) which contain invisible layout markup to help us identify critical sections (the layout itself is irrelevant, only the name is used, see Section 3.3.1). Besides, especially within WYSIWYG word-processors like *MS Word* it is no trivial task to keep this hidden markup intact and correct (see also in Section 3.4.2). We then convert these proprietary document formats in a first step to (flat) XML in order to be able to unify similar processes and to use standard concepts and tools as early as possibly. Finally, this intermediate document is converted to our schematic application XML document which conforms to a DTD (see Figure 3.16 for a small section) and thus can be validated by an ordinary XML parser.

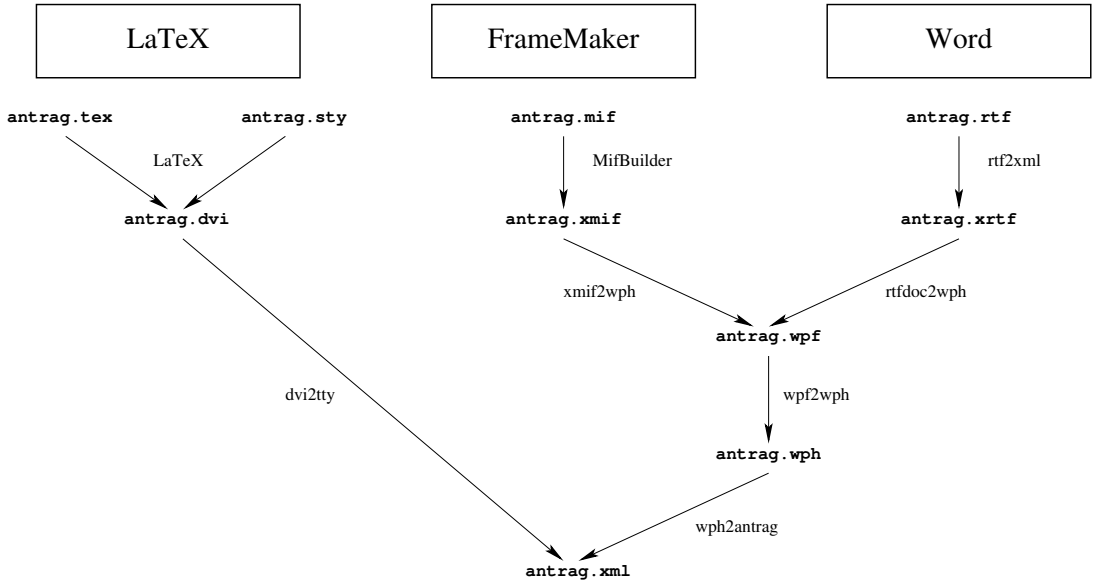


Figure 3.17: A scheme of the *Up-Translation* applied to submitted documents in the *GoldenGate* system.

Our document templates mentioned above as well as the used conversion steps during the necessary *Up-Translation* differ widely between LaTeX and *Word* or *FrameMaker* documents respectively (i.e. for LaTeX less and more simple steps are sufficient), therefore they will be discussed separately.

For documents created with the WYSIWYG word processors supported by *GoldenGate*, the *Up-Translation* process is as follows (see also in Figure 3.17):

1. *Unification*
   After having been uploaded, the electronic applications based on our templates (in *RTF* or *MIF* from *MS Word* or *Adobe FrameMaker* respectively) are interpreted by a special parser as well as some scripts and then translated into a unified XML-format. To be precise, an *RTF* file is first converted to an intermediate XML file using the script `rtf2xml.xom`[9] (by Rick Geimer) for *OmniMark*[10] (i.e. a programmed text-stream converter). A *Python* script is then used to further convert this into our so-called *wpfile* ("wp" for "word processor"). Such a *wpfile* is also the result when a *MIF* is converted in two steps by another *Python* tool which we implemented.

   The content of the *wpfile* is still very similar to the original markup in the word processor templates, but it is already independent from their format (to the extent that the same grant proposal either as *Word* document or as *FrameMaker* document will generate the same *wpfile*). In the final document only that metadata and layout information is retained which will be useful for further steps of the complete conversion process.

2. *Building the hierarchy*
   The *wpfile*, which still represents the sequential document model from the original grant proposals, is now transferred to a hierarchical model stored as XML which we call *wphiera* (for "word-processor hierarchy"). A *Python* script combines sequential detail information to paragraphs and these to sections which form the complete application. This is the first step were semantic knowledge about the underlying document is used (i.e. it is a grant proposal for DFG), for example to understand which paragraphs start a new section or which section ends the document.

[9] http://www.xmeta.com/omlette/rtf2xml
[10] http://www.omnimark.com

3. *Refinement*
   The final conversion step adds the semantic knowledge (which was already used in the previous step) to the document. We know (from the requirements for grant proposals to DFG) for example, that the first section contains "Allgemeine Angaben", so the previously generic elements of the XML document are renamed appropriately. The resulting XML file (the internal format of *GoldenGate*) represents the original document in form of semantic markup, which contains the structure as well as the content of the submitted grant proposal. This finishes our *Up-Translation* process.

For LaTeX documents the *Up-Translation* process is implemented somewhat different. The basic idea was to use the LaTeX processor (i.e. `virtex`) itself for the conversion. A special style-file (or macro-file, `antrag.sty`) delivered together with the template document (which also serves as example) enables the author to execute the translation locally in his own TeX-environment. This helps us to avoid the problems of different LaTeX-variants or the problem that the author could use macro-packages which are available to him only.

This procedure is possible because LaTeX-source documents already contain explicit markup (e.g. `\section{Geräte}`). The author is used to see it and to enter it. In most cases this is even semantic markup representing the hierarchical structure of the document. The single elements are commands (or macro calls) to be interpreted by the LaTeX-processor which means that we can modify them to suit our needs. So we simply defined own commands to mark the document elements which are to be extracted or modified existing macros respectively. Additionally, our style-file prepares two different modes of operation which are chosen by a simple switch at the beginning of the source document. Depending on the setting of this switch, LaTeX either creates normal layout for printing or it runs our new macros to generate a complete XML-alike document, finishing a major part of the whole conversion process. In the latter case, our new created or modified macros simply are responsible for the generation of correct XML-markup for all important content elements.

In a final step, the resulting document, which still is a binary presentation-oriented *DVI* file (the output format of the TeX-processor), has to be converted to the textual XML format. Up to now the document only looks like an XML file when printed or previewed on screen but it isn't one yet. The layout information stored within the *DVI* file is not of interest for us, so we can simply ignore it. For the extraction of the pure text we use the tool `dvi2tty` which offers exactly this function. After that, we run a script to remove superfluous whitespace and end our *Up-Translation* with a wellformed XML document which matches the result from the translation of a *Word* file as described above. See also the whole process in Figure 3.17.

For a presentation of results and a discussion of remaining problems as well as other experiences gained, read the following Section 3.4 below.

3.4 Results & Experiences

A prototype of the *GoldenGate* system as described above was implemented during the first year of the research project. Following our plan, we used an iterative approach, which means that we demonstrated the prototype to our project partners after each evolution step, collecting useful criticism, hints and recommendations as well as some acknowledgment. Very early we installed a version at the DFG department IID6 ("Computer Science") and let the people for whom the system was actually being built make their own tests and experiences, simulating their daily routing work. The results gained during this field tests lead to many improvements but also to extensions of the original system. Some of these were already mentioned above, others were not. Examples from both groups will be discussed in more detail in Section 3.4.1 below.

Additionally to these field tests, we presented the system twice to all department heads of DFG as well as some technical staff. The reactions were very positive from almost all sides, often resulting in the question "When can we have it?". The only (and sometimes harsh) criticism came from the IT group. These people were missing the "real database" (meaning complex and expensive, from a well-known company) and the future support for the system. We could easily answer to these issues: since Version 4 of the *Hyperwave Information Server* users are not limited to using the internal database, instead it is possible to connect for example an *Oracle* DB as storage component. As for the long-term support, both *Microsoft* and *Hyperwave* are companies that surely are not going to cease supporting their products in a foreseeable future. Additionally, we mediated a contact between DFG and a small but very engaged local company, who then offered an attractive support contract.

Other critical comments were spoken out from department heads of so-to-speak less "technical" sciences who were worrying that they will have to adjust their internal workflows to those of the Computer Science department when using our system. But this is exactly one advantage of our approach of building a system by combining standard components: this combination process can easily be adjusted to the needs of the customer, especially to slightly different needs of each member of a group of customers. Even more, it is easily possible to install an own (especially adapted) instantiation of the system at each department and to nevertheless combine these to a complete, distributed system. All we had to do would be to enable one feature of the *Hyperwave Information Server* that autonomously combines the databases of several distributed servers to one complete database hierarchy accessible from all participants.

All which can be said is that the final installation of *GoldenGate* for daily usage at our partners within DFG failed for "political" reasons. It was planned to do it, we were ready to do it and our partners were willing to do it, but it never happened, which was disappointing for both sides. As far as I can tell from the outside, reasons included the unwillingness to ignore the own IT group and other people who were planning their own electronic workflow system (producing lots of paper but nothing which could be used or even demonstrated). Additionally, the courage from the administrative side to try out something new and modern might have lacked. As I have heard, by the time when this dissertation is being written (i.e. several years later) DFG is still planning to "soon" install their own digital workflow system for the submission, the review and the management of grant proposals...

We made a last attempt when we presented and offered our system to the department for Library Sciences. Its head was open-minded to our ideas and after we adapted the prototype to their special needs (which by the way was as simple as we had expected) he let his staff decide democratically. The result was positive. Nevertheless, we never became the opportunity to actually install the system. One reason might have been, besides others already mentioned above, that the department head left DFG shortly after this decision (there was no coincidence).

Finally, we managed to get one real-life test-run of *GoldenGate*, at least as far as the authors' submission is concerned. Our prototype system (which was already extended and matured at this time) was used to electronically manage the proposals for Phase II of the research program V^3D^2 [Fel97]. The performance was very good and satisfactory, not only from our own point of view. See in Section 3.4.2 below for a detailed description.

3.4.1 Extension of the Prototype

In the previous sections it was already mentioned, that we had to extend the functionality of our *GoldenGate* prototype based on experiences gained from demonstrations or tests at the DFG department cooperating with us (e.g. see Section 3.3). Now one of these extensions will be described in order to illustrate our iterative approach and to show that our ideas have proven to be very successful. See also Section 3.4.2 for the results from our own real-life test.

When we installed the first version of our system at DFG, one of the secretaries used a newly arrived grant proposal (the electronic submission was not yet implemented at this time) as a test case trying

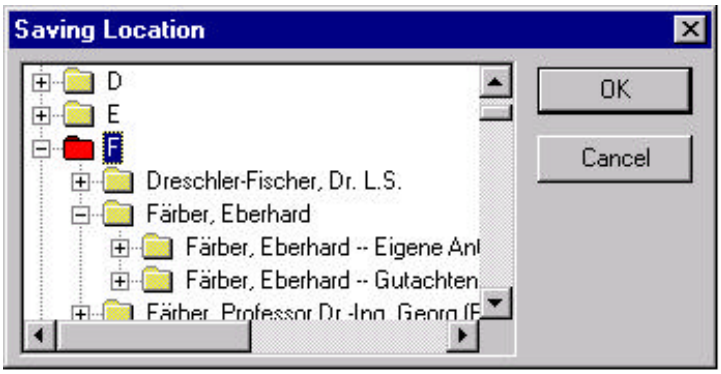


Figure 3.18: This is one of the dialogs which allow to choose the saving location in the *GoldenGate* database hierarchy for a document from within *MS Word*.

to perform her usual tasks with our system, while we were watching with big interest. After manually filling out the *dynamic view* (see Section 3.3.2) – a step that was automatized in the next version, which of course had been planned right from the beginning – she started *MS Word* and created a letter of confirmation for the applicant, entering most of the data she had just entered into the *Excel*-sheet again (like name, address, internal ID of the proposal, date of arrival etc.). We should have anticipated this, but as already said, no one could explain the complete workflow to us before and no written specification existed.

We clarified the situation in a long discussion and discovered that many similar documents were created during the life-time of a specific grant proposal, e.g. when the internal state of an application was changed or when results of the reviewing process were sent out, sometimes even to all proposals for one specific research program at the same time. A copy of all these letters was always kept together in one file with the original grant proposal. It became immediately clear and was the only logical way that we had to support this task within our system.

Due to our component-based approach the solution was convincingly simple: we had already a connection to the *MS Office* tools (through our *TCP/IP-COM interface*, see Section 3.3.2). So we easily added another function which extracts data from the database not for *Excel* or the Web browser but for the mail-merge feature of *Word*. We prepared a template for the letter in question and using some *VBA* code the data was received and filled into the fields for the mail-merge. The secretary at DFG (who was delighted when we presented this process later on) only had to print the document after possibly having made some special modifications. The other way round, the final *Word* document could easily be stored as another multimedia object in the *Hyperwave* database hierarchy, i.e. in the collection of the application it belongs to. While we were at it, we allowed any electronic document to be stored on the *GoldenGate* server from *MS Word*, by simple choosing the destination location within the hierarchy from a simple dialog we implemented (see Figure 3.18).

In a second step, we allowed this mail-merge functionality to be used arbitrarily. If someone then wanted to send a letter to one person or to a group of recipients (like all researchers in a program or from a specific field), he or she could simply choose the corresponding subtree from the hierarchy using one of several dialogs (see Figure 3.19 for an example). Again, the necessary data for the letters was extracted from the address database integrated within the *GoldenGate* system automatically. By the way, during this extension and improvement process we made one complete software package which was used by DFG at that time superfluous. The sole task of this package was to create mail-merges. With the help of our approach, several thousand lines of code were replaced by something which was already there (i.e. the appropriate functionality of *MS Word*) and just had to be used.

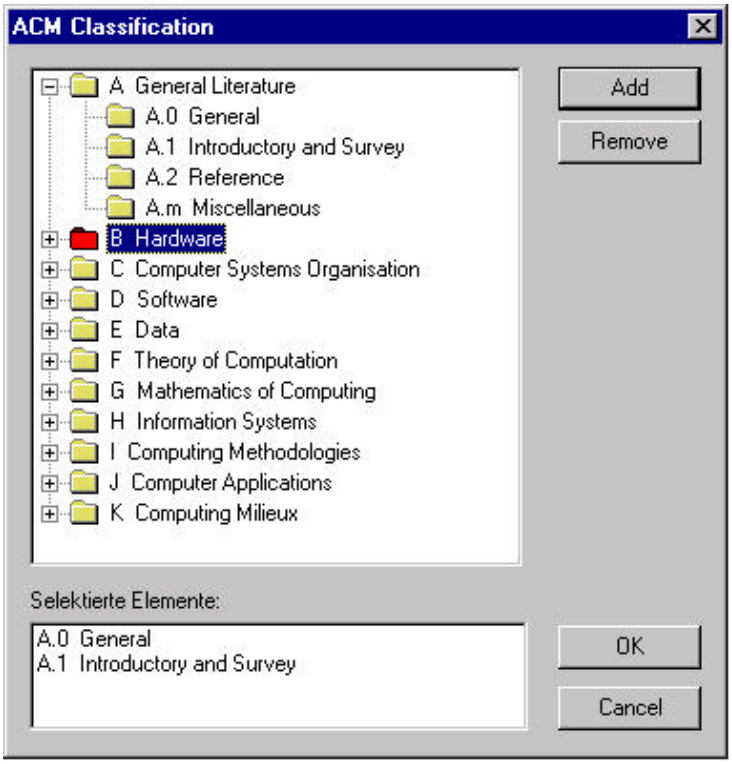


Figure 3.19: A dialog called from within *MS Word* to choose a research field based on the ACM Computing Classification Scheme.

The case described in the paragraphs above illustrates the most important lesson learnt during the *GoldenGate* project. Because of our approach to combine standard tools (i.e. Internet and office applications) with thin interfaces (see Fig. 3.4), the gain of flexibility goes hand in hand with a reduction of effort (i.e. time and money) for maintenance and adaption to new environments or needs, as compared to proprietary or existing commercial solutions. While the complexity of the implementation phase is similar to other systems, we have the advantage, that the actual amount of own "code" (including templates and interfaces) we have to maintain or to update is only a small portion of the whole product. The by far larger part (i.e. *Hyperwave Information Server*, *MS Office*) is maintained, improved and extended by other providers. This is exactly what we had hoped for, when we decided to follow this road (see in Section 3.2.3).

3.4.2 Real-Life Usage

When it became more and more improbable to get the opportunity for a real-life test within DFG (see Section 3.4 above), we decided to have a test-run on our own. Our cooperation partners at DFG (i.e. department IID6) agreed to receive print-outs of the grant proposals for the ordinary paper-based workflow from us (instead of getting these directly from the authors). As a result we used the *GoldenGate* system to cover the electronic submission of grant proposals for Phase II of the DFG research program V^3D^2 (see [Fel97]) which was about to start in late 1999[11]. The possible participants (most of them had already received funding during Phase I but several joined in for the second year) were encouraged to use our special *MS Word* and LaTeX templates (see Section 3.3.1) and to submit their proposals to the *GoldenGate* server, which we set up in our group. Figure 3.20 shows the homepage of our service

[11] http://graphics.tu-bs.de/v3d2

Figure 3.20: This figure shows a part of the *GoldenGate* system's homepage as set up for Phase II of the research program V^3D^2.

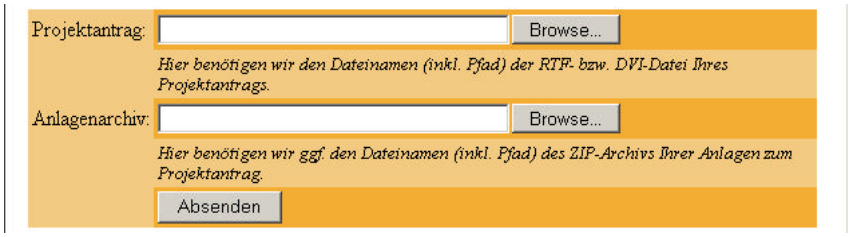


Figure 3.21: Here the essential part of the *GoldenGate* upload form as prepared for the research program V^3D^2 is displayed.

for V^3D^2 and Figure 3.21 displays the essential part of the form through which the submissions were uploaded. See also Figure 3.8 for the form to register as applicant. The important metadata was then extracted automatically and the appropriate *static* and *dynamic views* were generated (see Section 3.3.2). When the submission period was over, the electronic proposals were printed and handed over to the DFG department responsible. We kept the metadata (in the database objects and the *dynamic views*) and used it to generate the overview *Excel* sheet for the final committee meeting, already mentioned at the end of Section 3.3.2.

Except for one grant proposal (out of a total of 26) all documents we received were based on our templates, following our recommendation. We were in close contact with the authors to support them and to collect their experiences (and their assessment of the process, of course). Additionally we traced what happened during the XML Up-translation (see Section 3.3.3) process. As could be expected, the results were different for *Word* (9 submissions) and LaTeX (17 submissions) documents. Therefore, they will be discussed separately in the following paragraphs.

Word documents

We had already learnt very early during tests within our own group and in discussions with our partners at DFG that *MS Word* (as a typical example of an WYSIWYG word-processor) is not very "careful" as far as formatting styles are concerned. It is very easy to unintentionally delete or modify the given style of a paragraph or a character. By simply pressing backspace at the wrong position, by using automatic tools like the enumeration feature or by pasting text from another section the author often changes the styling of a complete paragraph. Usually, experienced authors can immediately compensate this by modifying the layout until it fits again (or by reassigning the correct paragraph style, which sadly is not a very common procedure). But, of course, this destroys our special hidden markup (because the author does not know anything about it and he should not as the idea is to achieve transparency), necessary for the automatic *Up-translation* of the grant proposal (see Section 3.3.3).

Our solution to this problem was based on the fact that we had already implemented GUI dialogs for many important elements in the formal data of the grant proposal (see for example Figure 3.7). We extended this approach and used the locking feature of *Word* to completely prohibit any manual modification of important sections in the document. Now the appropriate dialogs became the only way to change several data items and thus the hidden markup was safe. In a way, we reduced the possibilities of the authors (which in fact are already limited due to the strict guidelines of DFG) in order to improve the documents which they produce.

Of course, many authors didn't like that. This is understandable, because authors using WYSIWYG word-processors are "conditioned" to change any part of their document until it *looks* as they want it (which is not good, but this question is discussed in Section 2.1.2). This could also be put the other way round, authors did not understand or accept why we disabled the manual editing of some parts of the document (which was, after all, for their own good). As a result, it was not possible to automatically convert 4 of the 8 grant proposals submitted as *Word* documents which were created using our template. Obviously, the locking of sections didn't help much. Authors simply disabled the lock, changed some content (e.g. adding annotations) and thus destroyed our hidden markup. Some Authors finally even re-enabled the lock again. Maybe we should have anticipated this behavior from computer scientists but it only adds to the well-known problem of getting good structural information from a WYSIWYG word-processor. On the other hand, we should not neglect that 4 grant proposals were correctly formatted and structured and could be converted flawlessly into our internal *dynamic view*. So it does work, after all.

One problem we had with *RTF* files from *Word* is not a result of the authors changing the formatting styles but of the word-processor itself. *MS Word* can load and save *RTF* (which is no surprise, since *Microsoft* invented the format [Mic99]) but the more the document is edited and the more often it is written to disk, the more garbage is put into it. This ultimately lead to several cases where other tools, which usually are able to read in *RTF* perfectly, could not interpret these files any more. As Carsten Oberscheid put it in his Master thesis [Obe99] "When looking at this kind of RTF file, I don't know what is more surprising: that Word manages to create such files or that Word indeed manages to read them in again..." (translated from German by the author).

A common difficulty in data mining, to correctly interpret numerical values in free text, came also into play during our real-life test of *GoldenGate* (for both *Word* and LaTeX documents). Sure, thanks to our pre-tagging we had no problem to identify the right character strings, but additionally to the known problem of how to interpret times and dates, there obviously is also a vast number of possibilities to write amounts of money or numbers of people. For example the simple question "How much funding do you want for travel?" was not only answered with "4000" or "4.000,- DM"as one might expect but also with "4000 – 5000" or "viertausend". The necessary steps are clear: tell the authors exactly what format is expected (e.g. "use only digits") and check what they have entered. The latter is to a small amount possible with the LaTeX interpreter and it is very easy in a GUI dialog of the *Word* document which already uses *VBA* macros to process the input.

LaTeX documents

Generally, the grant proposals submitted as LaTeX documents performed much better. 17 were uploaded and 13 of these were converted completely automatically without errors. The other 4 documents contained minor errors (e.g. removing a command that must not be removed) which could easily be fixed and could have been completely avoided by more or clearer instructions in the template. Obviously LaTeX users are used to explicit markup and they tend to modify given templates only as far as is absolutely necessary (often this means only to replace example text with one's own words).

What did happen was for example, that content was added after the "proposal end"-command (i.e. `\end{ggAntrag}`) in our template (but before the `\end{document}`). This is no problem for LaTeX and the print-out looks good, but this results in a non-valid XML document containing content outside the root-element. As already mentioned, a simple instruction for the author (e.g. "Do not enter text below this line!") would probably avoid this situation.

The major problem that occurred with LaTeX documents (it did not prevent the *Up-translation* from working and thus could be considered a minor flaw) depended on the font encoding used. The original LaTeX fonts (*Computer Modern Fonts*) only contain 128 characters (in so-called `T0`-encoding), similar to the original ASCII charset (see Section 2.1.3). Other characters, in our case especially the German umlauts (äöüÄÖÜ), are then combined from several characters in a font with `T0`-encoding. Therefore they can not be easily extracted from a DVI-file. At least the `dvi2tty`-tool we used failed to do so. This resulted in wrong letters in extracted names or addresses.

One can think of several solutions to this problem. For example, `dvi2tty` could be extended to reconstruct such "multi-byte" characters. Another possibility is to use more modern TeX fonts with `T1`-encoding. These contain 256 characters, including the German umlauts. Of course, the problem would again arise as soon as the author wanted to use other languages with even other characters. Besides, many TeX installations did not include the newer fonts at this time. A general solution would be a completely Unicode-based TeX, possibly even in XML syntax (like *TeXML*[12] from *IBM Alphaworks*, which is not maintained anymore).

Summing up, I can confirm that the first real-life test of our *GoldenGate* prototype was a success. Especially the LaTeX templates worked very well and were broadly accepted by the users. Some minor problems (like having used a bibliography style which is not generally available) could be solved immediately during the submission period. Several solutions to the other problems (see above) were clearly visible and could have been implemented quickly, if the system would have come into further usage.

From the point of view of *MS Word* the situation was somewhat more difficult. We received only a small number of submissions in that format (surely due to the fact that the research programme was most of all for the Computer Science community) and nearly half of them was not automatically usable. On the other hand, the other half of the *Word* submissions worked well, which proves that our processing steps are functional. Additionally, authors were not very satisfied with the template (and especially its limitations), but we should have been able to improve this opinion with some modifications and more explanations. Even time might have helped us here, as for example the idea of structured documents is much more spread and accepted today as compared to 3 years ago.

What is interesting to note is that 2 years later, when the submission period for Phase III of V^3D^2 was about to begin, several researchers asked for the "electronic submission we used the last time". Obviously they had liked the system and wanted to use it again. Sadly, we could only tell them, that DFG preferred to build their own system which was not ready then (and isn't yet)...

[12] http://www.alphaworks.ibm.com/tech/texml

3.5 Summary

The project *GoldenGate* discussed in this chapter is the first of three example application scenarios in this thesis, dealing with the modern use of digital documents and Digital Library techniques. See also the following Chapters 4 and 5. *GoldenGate* focuses on the creation of and the access to "good" (i.e. structured, in a well-defined format, with metadata etc., see Section 2.1 above) during a specialized workflow process.

The *GoldenGate* project was started in 1997 as a cooperative research project of the German Research Foundation ("DFG") and our group. The goal was the design and the prototype development of a complete workflow for the electronic submission, the managing and the approval of research grant proposals at DFG. At this time, DFG interacted with the outside world online by paper documents and internally they used an integrated, proprietary database. Therefore, the need for a modernization (i.e. the application of a digital document and workflow management system) was acknowledged (see the introduction in Section 3.1).

An office workflow exclusively based on digital documents is now a good alternative but "paper" still has some advantages because it inherently offers well established methods for *access control*, *authentication* and *workflow*. These have to be modeled as accurately as possible by a digital workflow in order to be able to replace a traditional (i.e. paper-based) one, in addition to the many advantages which an electronic environment offers anyway, of course. Another important fact is that any system which implements the three key features mentioned can be used as an Information and Workflow Management System (see Section 3.2).

Hyperwave Information Server is such a system. It could be described as a Document Management System with a smoothly integrated WWW-gateway, meaning, all functions can be easily used through an off-the-shelf WWW-browser (see in Section 3.2.2). It is able to store generalized digital documents, ordered in (virtual) hierarchies, including bidirectional links (e.g. references) between them. Arbitrary metadata can be added easily. An included user management system allows for a fine grained access control mechanism. Other features useful in the course of the *GoldenGate* project are an embedded full-text search engine and the freely programmable layout templates.

Originally, there are two options to establish a digital workflow: either a new system (which then smoothly integrates into the existing environment, but the creation as well as the maintenance will be very expensive) can be implemented or an existing one (which reduces implementation and maintenance costs but the users will have to live with a compromise and the office processes will have to adapted to the software) is deployed. During the project we developed and implemented a third option, i.e. a combination of standard office tools and Internet technology as components with thin communication and interface layers. Such a system will fit perfectly into the users' existing workflow and offer a guaranteed functionality, because it is based on tools which they already *have* and *use*. It will only be necessary to maintain a small amount of program code, it won't be necessary to retrain the users and the system will evolve automatically if any of the embedded components is updated (by its manufacturer). See also the detailed discussion of this approach in Section 3.2.3.

The major components of an Information Management System (see in Section 3.2.4) are those for *input*, *output*, *access*, *storage* and *workflow*. In order to implement the *GoldenGate* prototype we took the software already used at DFG (i.e. *Microsoft Office*) for input, output and handling of documents as well as a *Hyperwave* server for the storage and the workflow and combined them using Internet technology (i.e. a WWW browser and TCP/IP-communication). Because the office tools are fully programmable, they can be easily customized to interact with *Hyperwave*. Storage and access control are basic features of the server and workflow (processes and use-cases) can be modeled by passing along the proper access rights for appropriate tasks.

Section 3.3 discusses the workflow for the management of grant proposals at DFG as well as our implementation in detail. As first step, a possible applicant downloads an application template for his

favorite word-processor. The resulting digital document is uploaded to the *GoldenGate* server, where it is converted to our internal XML format and stored in a special collection of the database hierarchy on the *Hyperwave* server. The officer in charge is informed and automatically generates a *dynamic view* (i.e. an *Excel* sheet to work with metadata important for the approval process). Additionally, *static views* (i.e. PDF documents) can be created acting as read-only documents and to keep snapshots of certain states (see also in Section 3.3.2). All documents remain stored in the database, of course.

The grant proposals are received in digital form and inserted into the *GoldenGate* system automatically as an internal XML document. But XML is (currently) not the format of choice for the end-user, who instead should be given the possibility to work with the applications they are used to (i.e. *MS Word* and LaTeX). However, the files created then contain mostly presentation markup, but we needed structural markup. Therefore a conversion step from a low level of abstraction (i.e. presentation) to a high level (i.e. structure) had to be applied, which is called *Up-Translation* (see in Section 3.3.3).

The four possible ways to achieve such a required *Up-Translation* are *explicit pre-tagging*, the use of *forms* or *text/pattern matching* and the analysis of *internal markup*. We implemented the latter. The documents from the supported WYSIWYG word-processors were first unified by converting them to a flat XML file and then changing the structure to become independent from the original format. During a second step, a hierarchical XML-structure was generated which finally was refined to our specified document type. For LaTeX documents we used the LaTeX-processor itself for the conversion: using a special style-file containing modified commands and macros a DVI-file was created, which already contains the XML-structure. This was extracted from the layouted document with the `dvi2tty` tool and in a last step refined to also match our document type. See the according section (3.3.3) for a detailed description.

The *GoldenGate* prototype as described was implemented using an iterative approach. It was regularly demonstrated and tested at our project partners which resulted in useful criticism, hints and recommendations. These allowed us to make many improvements and extensions. For example we added the functionality to store arbitrary documents and to automatically generate mail-merges. This was very simple and efficient, because we only had to use another feature of one embedded tool (i.e. of *MS Word*). This proves the claim of our approach to combine standard components, that the gain of flexibility comes together with a reduction of effort for maintenance and adaption to new needs (see in Section 3.4).

Unfortunately, the final installation of *GoldenGate* within DFG failed for "political reasons", though we and our partners were prepared and willing to do it. However, one real-life test run we could execute is described in the final section of the chapter, No. 3.4.2. We set up the *GoldenGate* server in our group to support the electronic submission of grant proposals for Phase II for the research program V^3D^2 in late 1999. The submissions based on our templates were uploaded as well as converted and dynamic and static views were created. Finally, the electronic proposals were printed and handed over to DFG. The LaTeX templates worked very well but with *MS Work* the situation was more difficult. All in all we proved that our processing steps are functional and that our approach was very successful. Many of the remaining minor problems could have been easily avoided with another iteration step of the prototype. Therefore, a lot of the experience gained and the techniques developed could be reused in our later projects *EG Online* and *MCP* (see the following chapters).

Chapter 4

EG Online

The *European Association for Computer Graphics*[1] (shortly *Eurographics* or even *EG*) is the major European-wide non-profit organization of researchers, students and professionals in the field of Computer Graphics (and No. 2 world-wide, following ACM's *SIGGRAPH*[2]). Quoting from Article 2 of the Association's Constitution: "The purpose of the Association is and shall be to contribute to and promote the advancement of Computer Graphics, primarily in Europe, by all suitable means..." [The02].

From very early *Eurographics* (founded in 1980) made use of the World Wide Web (e.g. the Domain Name entry `eg.org` was created in 1993) to present itself to the world and to deliver information to its members. During the late Nineties, the idea was born to extend these simple Web pages to a sophisticated online system offering Web services for distributed collaboration and administration to members, officers and others (see Section 4.1). This improvement was designed, implemented and administrated over several years within our group (and currently it still is), of course with the help and invaluable input from others.

This chapter will present the basic ideas, the history of advancement (see Section 4.1.1) and the present status of this group of new Web services, called *EG Online* (see Section 4.2). In the final Section 4.3, some interesting technical details of the implementation will be discussed in addition.

4.1 From Web Pages to Web Services

In the early days of the WWW (i.e. the early 1990s), all Web pages were purely static documents (see also Sections 2.3 and 2.3.2) in HTML-format (see in Section 2.1.5). They were more and more used by commercial or non-profit organizations (like *Eurographics*) and individuals to present themselves ("Who we are and what we do!") and to display information ("We have this to offer!"). During that time, the notion of Corporate Identity was transferred from other media to the Internet. However, these pages were only for reading and so they could be grouped into interesting and boring ones depending on how often their content was updated.

This situation changed dramatically with the invention of techniques (like *CGI*, *JSP*, *ASP*...) which allowed to create documents on the fly, at the request of the user, based on arguments supplied by the reader. The formerly static pages evolved into dynamic documents, where questions could be answered and actions could be taken, on request of the person running an ordinary Web browser. People having been only readers before could now truly interact with organizations and with each other. What is today called a "Web Service" was born (btw. this term is only recently being technically defined, see the Web Services Activity[3] of the World Wide Web Consortium).

[1] `http://www.eg.org`
[2] `http://www.siggraph.org`
[3] `http://www.w3c.org/2002/ws`

The first wide-spread Web Services were search engines and online ordering services (part of what is today called "e-commerce"). Their big advantage as compared to traditional services (i.e. mail order, telephone based or even personal) resulted from the nature of the WWW (i.e. an application on the Internet) as a distributed network with well defined document formats and communication protocols. Now, anyone could use such a service anytime from (nearly) anywhere in the world, even concurrently. This possibility of easy distributed collaboration made the new technique interesting for an international non-profit organization like *Eurographics*.

4.1.1 Short History of EG Online

A Domain Name entry for the *Eurographics Association* (`eg.org`) was registered in 1993 and shortly after, the first Web server (`www.eg.org`) was started in Manchester, UK. At the same time, a new position within the *EG Executive Board* was installed, a so-called *Online Officer* (later: *Online Board Chair*). This job was taken by Dr. Ivan Herman who prepared the first Web pages and also managed other "online activities" like emailing-lists (using `majordomo`[4]) remotely from his office in Amsterdam in the Netherlands. Thus, to a small extent *EG* was already then taking profit from the possibilities of distributed collaboration offered by the Internet.

During the early years of *EG Online* (as it was later called, when it truly became a set of Web services) the main task was to deliver information about *Eurographics*, the Association's activities and its publications to members as well as to the public. Several parts of these first contents can still be found (updated to the present situation, of course) in the current system. For example the pages which today describe the major activities and the internal organization[5] are strongly based on the original documents from that time. Later, Ivan Herman also made the first steps towards a real service when he added an online ordering form for *EG* publications and the possibility to upload documents for the *Executive Board* and the *Executive Committee* respectively.

Using the publication order form a user could select one or several publications and the way of mailing requested. After clicking the `submit` button a server-side script, written in *Perl* [WCO00] and accessed via the *Common Gateway Interface*[6] (*CGI*), computed the total cost (incl. shipment) and presented a nicely formatted document in the user's WWW browser. This could then be printed and faxed to the Association's Secretariat for further processing. The document upload as a second service was implemented similarly, again using HTTP, CGI and a Perl script, which just received the document file and stored it in a special directory of the Web server's filesystem.

In 1998 *Eurographics* decided to try out whether more of the routine work could be automatized and whether more services could be offered to officers and members using Web technology. Of course, being a non-profit organization with limited resources it had to be a low cost solution and thus buying the necessary technology (or the complete services as such) was not an option. The first idea was to support the administration of the Association (namely the work of the Secretariat) with a Web Service. At that time this was still seen very much separated from the already existing Web server.

Therefore the first online application which was designed and implemented in our group was the *EG* membership DB. Up to that time, this DB was simply a collection of files (e.g. some spreadsheet tables) stored on the personal PC of the Association's Secretary in Geneva, Switzerland. This was a bottleneck since the complete membership administration like the addition of new memberships or membership renewals, the removing of resigned members and the checking whether all fees were paid correctly (for approx. 600 members worldwide) had to be done by the Secretary (who had also a "real" job) in his spare time. Additionally, other officers (in other countries, like the Treasurer or the Chairman) had to interact with or needed information from this database. So it was a promising but also a somewhat challenging

[4] http://www.greatcircle.com/majordomo
[5] http://www.eg.org/EG/Organization
[6] http://hoohoo.ncsa.uiuc.edu/cgi/

task to enable the management of and the access to the membership DB through a Web Service.

Based on our experience from the *GoldenGate* research project (see in Chapter 3), which was running at that time, we decided to use a *Hyperwave Information Server* (*HIS*) as Content Management System and Web server for the new *Eurographics* system. See Section 3.2.2 for a detailed description of *Hyperwave*'s abilities and characteristics. Especially the integrated database and WWW interface as well as the user management and access control features of *HIS* proved to be very helpful.

The new membership DB was a success (see below in Section 4.1.2). As a consequence, the next step was taken very soon: we merged the former Web pages and this new service, forming *EG Online*. Both, the membership data and the information documents were now managed and delivered by our *HIS*, located in Braunschweig, Germany (respectively Bonn, before our group moved). After that, the system was more and more enhanced with more services (like for example the *EG Digital Library*) and better features (e.g. the online membership application & renewal form). See Section 4.2 for a thorough description of the present situation.

The popularity and (as a result) the usage of *EG Online* grew continuously during the last years. It quickly reached the point when we couldn't tolerate the additional load on our *Hyperwave Server* (which originally was meant to be used for research exclusively, see for example Chapters 3 and 5) anymore. The solution was simple: *Eurographics* bought an off-the-shelf (though state-of-the-art) PC and from that time on, the Web service has been running on a dedicated server. Later, a second PC was added to act as a mirror and fallback solution in case of a disaster, because the every day work of *Eurographics* became more and more dependent on the availability of *EG Online* (see the technical details in Section 4.3). Figure 4.1 shows a snapshot of the current entry page to the server, indicating already many of its functions.

EG Online, the online service of *Eurographics*, has not yet reached the end of its evolution (and possibly never will). Each new chairman of the *Online Board* added new impulses. For the future, our group is cooperating with the current *Online Board Chair* (i.e. Prof. Charles Wüthrich from the Faculty of Media, Bauhaus-University of Weimar, Germany) to improve both the presentation as well as the structure of the information and services offered. A first prototype has already been made available internally. The present discussion is centered around the question whether to upgrade the system to a new and more powerful technology as a followup to *HIS*.

4.1.2 Results, Experiences and Extensions

Today *EG Online* is a broad collection of Web services, a source of information, a place for the presentation and an important tool in the daily work of the Association. The services offered to members, officers and partly also to the public range from the membership database (including individual homepages and forms to start or renew a membership) over document archives for boards and committees to and not ending with a *Job Center* (for job searches and offers). A complete list of all services including detailed descriptions can be found in Section 4.2 below.

The experiences with the system are very positive from both sides, i.e. members and officers of *EG* as users on one hand and our group as implementers and administrators on the other hand. This is the major reason why the number of services transferred into an electronic workflow has constantly been increased. The users like the idea of efficient, distributed collaboration which is typical for a European non-profit organization with members from all over the world and which becomes very simple using WWW techniques (i.e. no special hard- or software is required). In our group, we are very satisfied with the performance of the *Hyperwave Information Server* as far as the new implementation and the administration of the Web services is concerned. Until now all tasks which were successively added could be easily implemented in a few days using the servers internal features (see Section 3.2.2) or by adding *Python* [RJ99] scripts run as *CGI* programs or regular jobs ("daemons"). All in all, the idea of a Web Service is a valuable step forward.

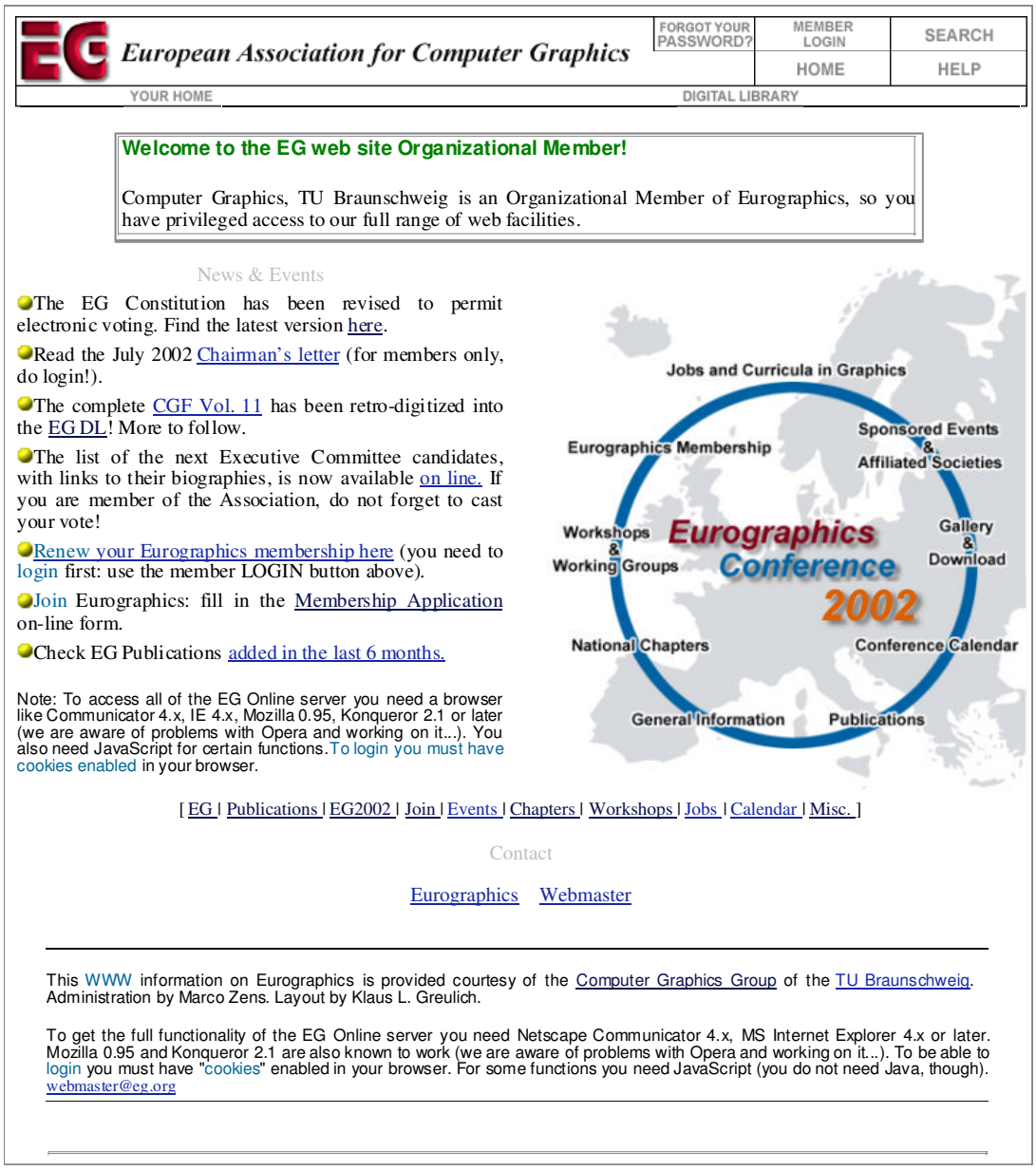


Figure 4.1: This Figure is a snapshot of the current entry page to *EG Online* at `http://www.eg.org`. Many of the (external) services offered are indicated.

The history of *EG Online* is a story of constant extension (see the previous section). The first application (i.e. the online membership DB) was an experiment which became successful. The online database stored the membership data and allowed to query as well as to modify it through a Web interface. The people working with this DB (i.e. officers of *Eurographics* like the Secretary and the Treasurer) considered it useful and immediately came up with new ideas how it could be even more helpful. For example, a member's entry could contain his or her email address, so it was a logical extension to update the mailing list `members@eg.org` automatically, whenever this address changed. The solution was quite easy: since then, a script is run every night which queries the *HIS* for changed email addresses and then updates the lists of the *EG* `majordomo` (located on another server) via administrative emails.

The first major improvement of the online membership DB was based on the idea, that members should update their data (at least the "public" part, like name, address etc.) themselves instead of having the Secretary doing it. So each member's dataset was split in a "public" and in a "private" part (i.e. type of membership, date of last payment...) and accounts were generated for each member which allowed them to edit the appropriate part of their own data in a modified Web form. In a next step, members were allowed to upload images as well as CVs to give their "homepage" a personal touch. From editing a membership dataset it was not a very big step to automatically creating or renewing a membership through another new Web Service. One of the latest additions to the group of membership services is a function which allows members to retrieve a hinting phrase for a forgotten password or even to reset the password. To be continued.

Another complex Web Service recently used by *Eurographics* and offered to partners is the *Managing Conference Proceedings* system (*MCP*) [FZ01]. We developed *MCP* in a research project as part of the *Global Info* programme[7], see Chapter 5. In short, *MCP* is a complete electronic managing and publishing process for publications of conferences, workshops and similar events. It supports the input (i.e. formats, templates and data-mining), the managing (i.e. submission and review) and the output (i.e. cross-media publishing, pre-press tools) of "papers" as well as secondary tasks (e.g. automatic reminders). The first prototype was used for the Full Paper submission and review at the *Eurographics 2000* conference and the Association has been using updated versions ever since for its yearly conferences and for several workshops. Though *MCP* is not directly linked to *EG Online*, it is however based on similar ideas and techniques, having taken benefit from previous experiences and it has been run on *EG*'s server several times successfully.

I'd like to share one other experience. When starting to develop and to run Web applications one sooner or later will reach the point when other people's personal data is gathered, processed and possibly even stored. Even worse, this private data is transmitted over open channels (i.e. the Internet) and stored electronically. Therefore it will become important to consider security measures like password protection, encryption and secure authorization. This is important not only for one's own good and for that of the customer/user but also because today many countries (e.g. Germany) have special laws for the protection of data privacy on computers and within networks. For example the European Commission has pointers[8] to national bodies behind such laws and to background information. Usually laws like these guarantee the individual that only the absolutely necessary part of his/her private data is stored for the least necessary amount of time and not given to third parties without requesting permission first etc. For *EG Online* the German law applies, so we have to get a written permission from every member that we may store his membership data electronically. We have to delete it upon request and automatically when the membership is canceled.

[7] http://www.global-info.org
[8] http://europa.eu.int/comm/internal_market/en/dataprot/

4.2 EG Online Today

Through its current *EG Online* system at http://www.eg.org (see also Figure 4.1) the *Eurographics Association* offers, besides the usual Web pages for information and presentation, many Web services to officers, members and other people. The system including the internal database and the WWW front-end has been developed in our group. We are also responsible for the management and the administration. When this state was reached in the year 2000, *EG* created a new position within its *Online Board* named *Deputy Chair (of the Online Board)* which is since then filled by myself. As such, now an overview of the complete system will be given, followed by a more detailed discussion of several interesting and/or important services. For some more technical background see Section 4.3.

To the anonymous user with an ordinary WWW browser, *EG Online* presents itself as a set of Web pages with an uniform layout (a so-called "corporate identity"). The homepage contains, besides an imprint and a manually updated list of current news, links to the important sections, with most of them each representing one the services described in the sections below. The Web services not discussed explicitly in the following sections 4.2.1 – 4.2.6 include the conference/workshop support, the external sections and the electronic voting system.

EG's support for conferences and workshops consists of the offer to host WWW homepages and mailing-lists within the infrastructure of *EG Online*. Additionally, announcements can be sent through the Association's own mailing-lists and news sections. As an example, the WWW pages for the *Eurographics 2000* conference are placed on our server and can still be accessed there[9]. The latest addition to these services is the support for the production of proceedings through the *MCP* system, which is discussed in Chapter 5. *MCP* was used for the *EG* conferences 2000 – 2002 and for the workshop on *Virtual Environments* in 2002.

The external sections are made up of collections were *National Chapters* and *Working Groups* or special boards of *Eurographics* can place their own content. They can administrate and control it individually through their browsers using the default features for document management of *Hyperwave Information Server*. But still, documents placed in these collections are automatically displayed using the same layout as the other sections of *EG Online*.

The final service to be mentioned here is the new electronic voting system of the Association (*E3G* for *Electronic Election for Eurographics*). In the near future, this will allow members to cast their votes for the *Executive Committee* electronically instead of returning the ballot paper by mail. A prototype of the system was demonstrated already and the constitution has been updated to allow for a non-paper based election.

One of the key features to the usability of *EG Online* are the context sensitive menus usually located in the top header of every page (see for example in Figure 4.2). These menus offer different functions depending on the current login status, on the specific access rights of the user (which in turn depend on his "function" within the Association) and on the actual location within the information hierarchy. For example an anonymous user only finds the usual buttons like LOGIN or SEARCH whereas the Secretary (after having logged in) can also click on buttons which take him directly to the membership DB or to the document archives of the *Executive Board*. When looking at the document archive, new buttons to upload or to delete a document are shown. For every member a button YOUR HOME is created, which takes him or her directly to his/her own membership dataset (see below).

4.2.1 Membership DB

The online *Eurographics* membership database was the first one and also the basis for most of the other services which today form *EG Online* (see in Section 4.1.1). It has been very much extended since the first implementation but the general structure is still the same.

[9] http://www.eg.org/EG2000

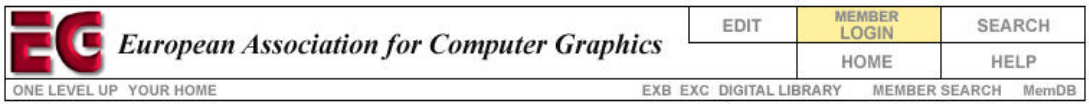


Figure 4.2: This is an example of the context sensitive menu in the default layout of *EG Online*. The `EXB`, `EXC`, `MEMBER SEARCH` and `MemDB` buttons are only available to users with sufficient access rights.

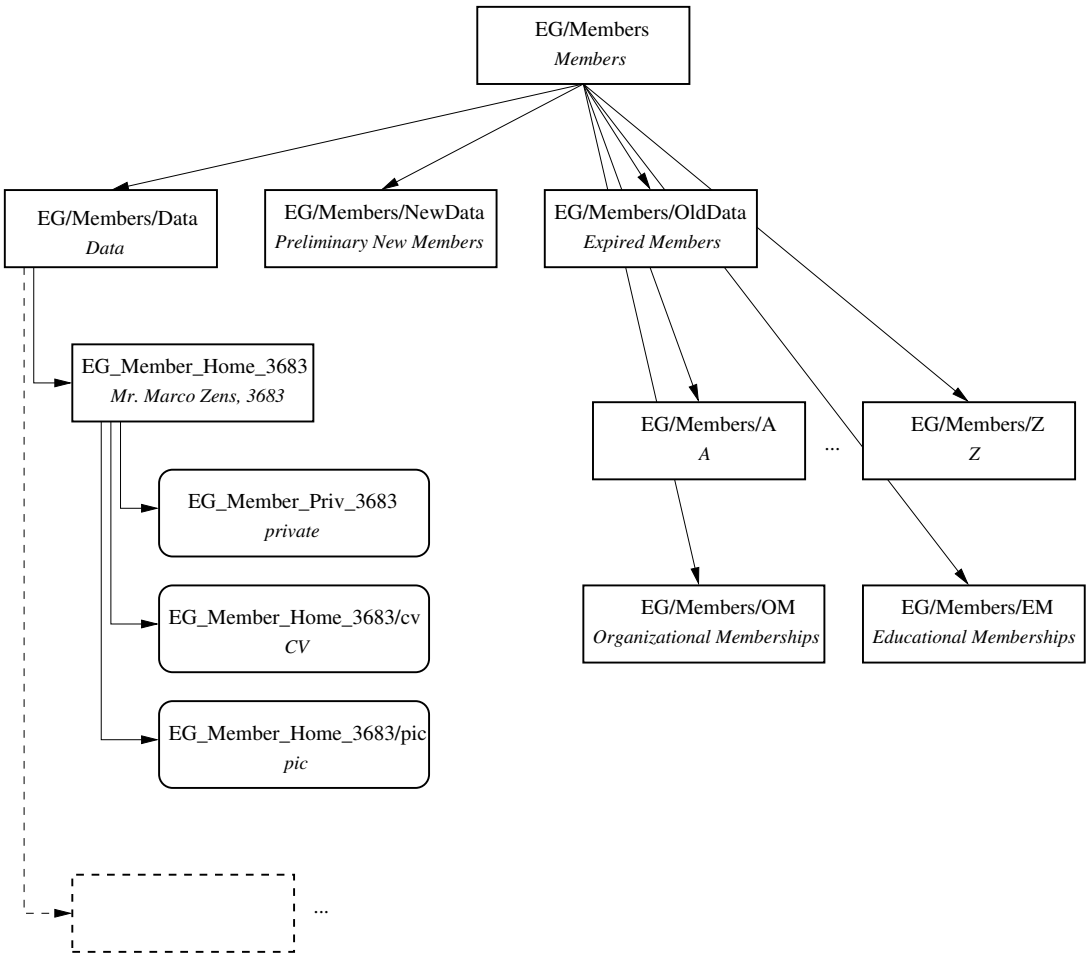


Figure 4.3: This Figure presents a sketch of the hierarchical structure of the membership DB of *EG Online*. The collections `A` – `Z`, `EM` and `OM` contain references to the objects in `Data`.

As can also be seen in Figure 4.3, the database contains one unique collection for each active member. This collection contains one data object and optionally two documents, an ASCII or HTML document (i.e. a CV) and an image of the member. These additional documents can be uploaded or replaced by the member through the personal homepage (see also the example in Figure 4.4) whenever he or she wants to. Several attributes of the collection itself store the personal data of the member, the so-called "public data" to be precise. This is metadata like name, street or email address which can be modified by the member himself and which is displayed to every user on this member's homepage (i.e. the collection). Of course, the image and the CV are also always displayed, if available.

The mandatory data object in each member's collection serves as a well-defined place to store the so-called "private data". Therefore the access rights for this object are set as such, that only a very limited number of officers of the Association can read it (like the Chairman) and even less can change it (i.e. the Secretary and the Treasurer), excluding the member who has only read access himself. The "private data" mainly consists of information about the membership status (like the type of the membership, when it will expire or when the last payment was made). An important entry is the field storing the "function", i.e. the position or role of this member within *Eurographics* (e.g. Chairman, Publication Chair, Fellow or ordinary member). Again, both the public and the private data can be changed online through WWW forms, but only by people with the proper access rights (see the links on top of the page in Figure 4.4).

All the information in the membership DB is not only stored and can be edited, instead it is also used to control access and to update other services. For example, the email address in the public data is used to update *EG*'s mailing lists (see Section 4.2.4) and the personal access rights are computed from the function field together with the membership type in the private data. For example, a member of the *Executive Committee* is granted access to the respective document archives (see Section 4.2.3), a member with membership type "electronic subscription" can read all the material from the Digital Library (see Section 4.2.5) and so on. Whenever an entry changes, this internal organization is kept consistent either immediately (using integrated features of *HIS*) or over night, when a couple of scripts to do these kind of jobs is run regularly (like the one updating the mailing lists).

It is important to keep all this data accurate and up-to-date, which can be best done by the members themselves. Therefore they can get to their page easily by a link (`YOUR HOME`) from the *EG* homepage where they are offered intuitive and user friendly functions to update their data, simply using their browser. Especially new members logging in for the first time are welcomed by a text-box on top of the homepage which asks them to go to their personal data and to update it as well as to change the initial password (which can also be done by simply clicking a link on the personal page). Another script which is run at night informs the Secretary of possibly interesting changes by email. This is a big improvement offered by this Web service compared to the previous years, where members had to inform the Secretary in order to have their data updated (what they rarely did).

Of course, the membership DB is not only a storage but also a source of information for members (e.g. the tables displaying the people in the *Executive Board* and the *Executive Committee* are generated dynamically from the objects in the database) and officers (e.g. the Secretary can dump all important data in order to prepare a report on the membership status). To get the information required, users with sufficient access rights can browse the membership DB either alphabetically (via a virtual hierarchy, see Figure 4.3) or by group (all members, organizational memberships, recently expired memberships...). Last but not least, they can use the search function by clicking on the `MEMBER SEARCH` button (another context sensitive feature, only offered when someone with proper access rights is browsing the member section). The search form offers fields to query for name or membership number and/or country. The result list does not only present the matching members but it gives also a short indication whether the membership is currently active or not. This feature is used by organizers of events who want to check whether a claimed reduced fee because of an *EG* membership can be granted. Of course, the member search is implemented by using the internal search algorithms of *Hyperwave* to check the respective fields in each member's data objects (see above).

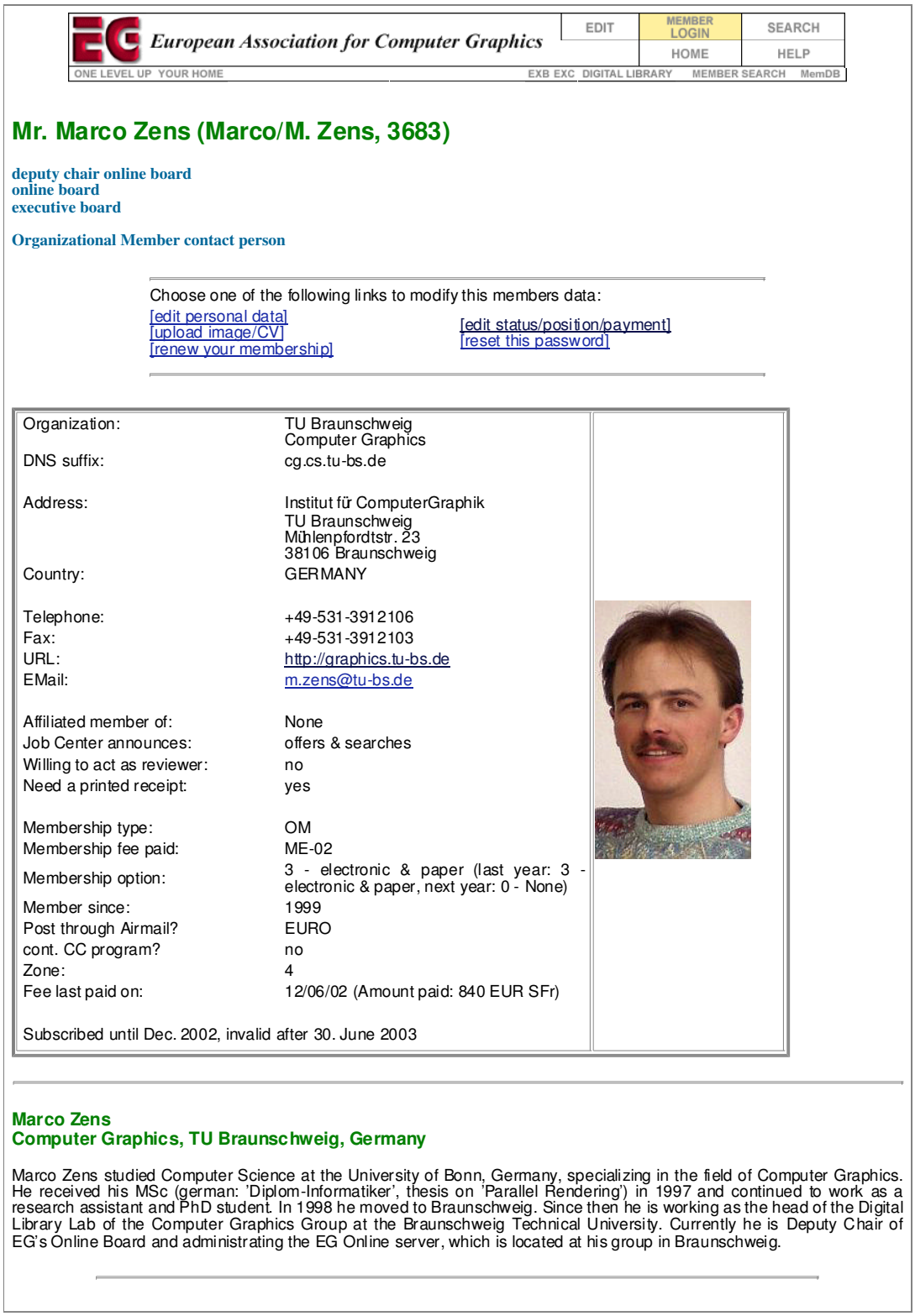


Figure 4.4: This is an example of an *EG* member's homepage, with all data displayed (i.e. someone with full access rights is logged in) and a CV as well as a picture available.

In the first year of *EG Online* (and before) people who wanted to become an *EG* member or who wanted to renew their existing membership could only download a form, print it, fill it out and send it to the Secretary by ordinary mail or by fax. The Secretary had to update the online DB manually or by running a script to create a new membership respectively. Now, for both tasks WWW forms are available. The renewal form is pre-filled with data from the database when the member is logged in (i.e. identified). When submitted, a *CGI* script is run which updates the database entry, sends an acknowledgment email to the member and informs also the Secretary by email. He can then approve the renewal as soon as the payment of the new membership fee is confirmed by simply clicking a link on the member's homepage.

A similar procedure is started when the new-membership form is submitted. The Secretary is informed by email and a new membership collection containing the entered data is created immediately. This new collection is stored at a special place (i.e. the collection for "preliminary memberships", see in Figure 4.3), meaning it is not yet active. Only when the necessary payment is received (by bank transfer, credit card or sending a cheque) will the Secretary or the Treasurer activate this new membership by clicking a link on the member's homepage. Then, a *CGI* script moves the collection into the hierarchy of active memberships and sends an email containing access information (i.e. username & password) to the new member.

At each first of a month, a script is run automatically which checks for expired memberships (by querying the "Expiry" field in each member's private data). These memberships are then made inactive (the accounts are deleted) and they are temporarily moved into another collection ("Expired Memberships", see Figure 4.3). Here they are kept for a while because often members just have forgot to renew their membership. In this case, the inactive membership can be made active again ("revived") by simply running a script. An additional script can explicitly delete a membership, if a member resigns.

A very new service in the membership section is activated by clicking on the button labeled `FORGOT MY PASSWORD` in the header of the default page layout. Until recently, the Secretary had to deal with these cases (happening quite often) manually. Now, each member can enter a "hinting phrase" into his data object. When someone presses the button mentioned, a form is displayed asking for a membership number. When submitted, an email containing the appropriate hinting phrase is sent to the email address of the member with the membership number entered. If no hinting phrase for this member is available, the password is reset to its initial value and the member receives an email telling about this.

4.2.2 Publications

One of the services within *EG Online* already mentioned (since it had also been available on the previous WWW server, see Section 4.1.1) is the publication section. First, it gives an overview and information about the books and journals published by or in cooperation with *Eurographics*, namely the *Computer Graphics forum* journal, the conference and workshop proceedings as well as an older book series (see also Section 4.2.5 about the Digital Library below). As a second part, this section contains documents, guidelines and examples which describe how to become an author of an *EG* publication and what to do when you are asked to review an article submitted to the journal.

The third and most important part makes up the real service: the *EG* bookstore. Technically this is a database which stores metadata about all publications that can be ordered directly through the Association by members as well as non-members, plus an online order form and, last but not least, a script to add new books. For each new publication, this script creates one object in a special collection of the *HIS*. These database objects contain the information necessary for the order form like internal ID, title, authors, price, weight etc. Additionally, an HTML document can be added, for example an abstract or a table of contents. When a user accesses one of these objects with his browser, the metadata is dynamically displayed as a table on top of the page by the layout templates, together with a link to the respective object in the Digital Library (if available) and the HTML document (again, if available).

Several virtual hierarchies (with references to the original data objects) allow the easy access to and the browsing of all publications, e.g. alphabetically by title, by kind of publication or by date. Of course,

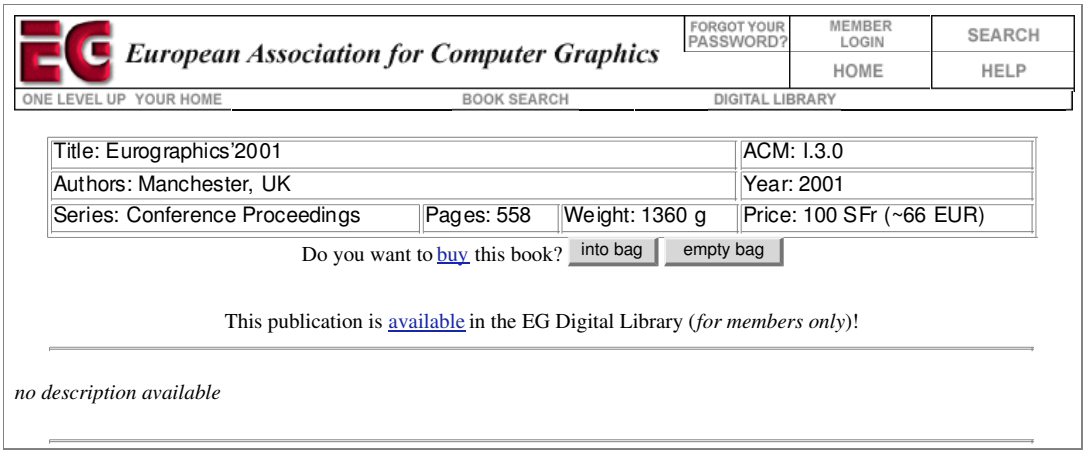


Figure 4.5: An example of one book in the *EG Online* publication database. The metadata is displayed, additional information is not available in this case.

a search function is also available. When a user is browsing within the publication section of *EG Online* a button `BOOK SEARCH` is added to the menu heading of the page layout. When clicked, a search form (based on the default search form of *Hyperwave*) is presented, being specially designed to query for titles and authors within the publication database.

For the order form the well-known "shopping basket" metaphor has been implemented by using session cookies in the client's browser. By simply clicking a link (see the example in Figure 4.5), every object in the publication database can be added to the order list. When a user continues to the order form (which lists all available publications) these objects are preselected. The customer only has to add his name and address as well as the kind of shipment requested. When submitted, a *CGI* script started will compute the total cost (based on the price and the weight data stored in the database objects) and display the final order which can then be printed, signed and forwarded to the Association's Secretariat. Some publications are sold through the Association only to members. The service takes care of that, because the order form offers those books only when the customer has identified himself as a member by logging in. For non-members, a hyperlink to the publisher who actually sells the book to the public is displayed.

A final function of the publication section which is quite useful but was very easy to implement is the list of recently published books (currently these of the last six months). This is simply an instance of *Hyperwave*'s feature called *stored query*. A search querying for all objects within the publication database which were created in the last six months has been stored and can now be restarted by clicking on the respective hyperlink. This link is available from the publication section as well as from the *EG Online* homepage, allowing a very quick check whether a new, interesting book is available.

4.2.3 Document Archives

Similar to the publications section (see 4.2.2) of *EG Online*, the Document Archives are a Web Service which was already available on the previous WWW server (see in Section 4.1.1) at least in a very simple form. Within the new system, its functionality has been enhanced to a great deal by taking advantage of the user management and the fine-grained access control (provided by the membership DB services, see 4.2.1) as well as of *HIS* features like creating and deleting database objects through the browser.

The usual access to the document archive section is for reading. Technically, the archives are ordinary *Hyperwave* collections in a hierarchy, so they can be browsed and read or documents can be downloaded using an WWW browser. See Figure 4.6 for an example of one archive. More functionality is provided through a set of buttons in the menu-bar (`UPLOAD DOC`, `DELETE DOC`, `CREATE ARCHIVE`), appear-

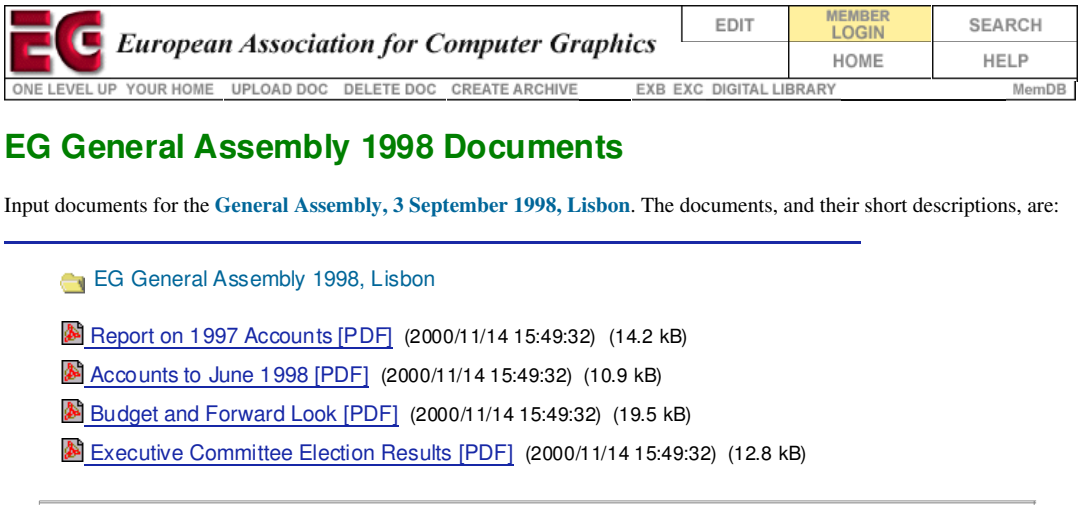


Figure 4.6: An example of a collection in the document archive section of *EG Online*. See the buttons in the context sensitive menu-bar.

ing depending on the context. These buttons are displayed if the user is logged in, if he is within the document archive section of the system and if he has write access there. The "document buttons" are only offered if the user has entered a specific archive.

Every user with write access in the current collection (within the archive section) can create a new archive in it by simply clicking on the "create" button. After that, a form is displayed where he has to enter a title and the type of the archive to be created. Four types, corresponding to different access rights are available: "Public", meaning *EG* members can write and everybody can read, "Members only", i.e. reading and writing by members, "EXC only" where reading and writing is only granted to members of the *Executive Committee* and "EXB only" respectively for the *Executive Board*. When the form is submitted, a *CGI* script is run which creates the new archive.

Documents can be added with the `UPLOAD DOC` button which displays another form to enter the filename and the title of the new archive document. The file is then uploaded and put into the archive. Whenever a new document is inserted into the "EXB" or the "EXC" archive, the members of the respective board are informed by an email sent to the appropriate mailing list. Last but not least, the "delete" button can be used to remove a document from an archive. It is important to note that a user can only delete those documents which he has uploaded himself.

4.2.4 Mailing Lists & Aliases

The *Eurographics* mailing lists and email aliases are no Web Services in a strict sense and they are only loosely connected to *EG Online*. On the other hand, the email services are controlled and used by several services of the online system and they are also hosted by our group. Therefore they will be discussed here shortly.

Eurographics maintains several email lists for the public (i.e. `general@eg.org`), for its members (`members@eg.org`) and for boards or officers (e.g. `exc@eg.org`, `secretary@eg.org`...). Additionally, there are some more simple email aliases for virtual addresses that map to one or a small group of person which don't change frequently (e.g. `webmaster@eg.org`). The "internal" mailing lists (i.e. all except for `general`) are updated every night by the membership DB service (see Section 4.2.1). Similar to the access rights on the server, the assignments to the mailing lists are computed from the "function" field in each member's private data (in most cases by a simple one-to-one mapping). This

keeps all the lists consistent and the included email addresses up-to-date. Because we use `majordomo` to run the mailing lists, this is very simple: when finished, the script computing the assignments sends an email to `majordomo@eg.org` containing all necessary administrative commands (like `subscribe` or `unsubscribe`).

One special case is the public mailing list `general@eg.org`. "Public" here means, that it is open to everybody. It is used by *EG* to distribute announcements regarding own (or in-cooperation) events like the conference and workshops. As with any email list hosted by the `majordomo` software, people interested can simply subscribe to it by sending a `subscribe`-email to the system. In this case, we have added a more easy way by providing a WWW form[10] within *EG Online* which takes an email address and runs a *CGI* script. This script then adds the address to the list by sending a short administrative email like the ones already mentioned above.

4.2.5 EG Digital Library

One of the most important parts of the services which *Eurographics* offers through *EG Online* is the *EG Digital Library*[11]. Simply speaking, the *EG DL* is an online collection of electronic versions of *EG*'s publications, i.e. the *Computer Graphics forum* journal as well as conference and workshop proceedings. See Figure 4.7 for a snapshot of the homepage.

All available papers are stored as PDF. But the *EG DL* is much more than a simple digital collection of printed publications. Also tables of contents, abstracts and so-called multimedia attachments are available, the latter being images, videos, audio or other digital documents following the "Generalized Documents" paradigm (see Section 2.1.1) which are submitted by the authors and which are not present in the printed version, of course. Additionally, also this digital library service makes use of the features of the *EG Online Hyperwave Information Server*. For example a search function is offered, taking advantage of the integrated full-text search engine (*Verity*). Also the presented material and the kind how it is displayed is depending on the user who is currently logged in (see below). Last but not least, all pages in the *EG DL* are displayed using a certain layout template, mostly identical to the general *EG Online* layout (e.g. using the same context sensitive menu mechanism and so on).

In more detail, the general structure of the *EG Digital Library* is as follows: four main sections contain (the electronic versions of) the journal (*CGF*), the *EG* conference proceedings, proceedings of *EG* workshops and the *EG Bibliography Database*, see also Figure 4.7. The journal section falls into the volumes (currently 9 – 21), each containing four or five issues respectively. There is an additional collection for currently accepted but not yet published material. Thus, digital subscribers to *Eurographics* get the benefit of being able to read new material before it is even printed.

The collections for the issues contain the actual papers (as PDF), metadata (e.g. title, authors, abstract), a table of contents, and additional multimedia documents (if available). Here the feature of *HIS* that links between stored documents are first-class database objects on their own is used heavily. The server outputs these links through the Web interface only if the current user has read access on both the anchor and the source document of the hyperlink. As a result, using the same URL and being shown the same document (e.g. a table of contents) two users with different access rights automatically see different content. This is illustrated in Figures 4.8 and 4.9. In both cases the Web browser displays the same document (i.e. the same URL) but in Figure 4.8 the user is not identified ("anonymous") and therefore he can only access the table of contents and the abstracts, as it is the current policy of *Eurographics*. While creating the snapshot in Figure 4.9 on the other hand, the user has identified himself as a member with digital subscription and thus all hyperlinks are inserted, meaning he can access every document of this collection (see also the listing of the collection's contents below).

Eurographics offers four levels of access to its digital library (see also Section 4.2.1). As already

[10] http://www.eg.org/EG/Docs/register.html
[11] http://diglib.eg.org

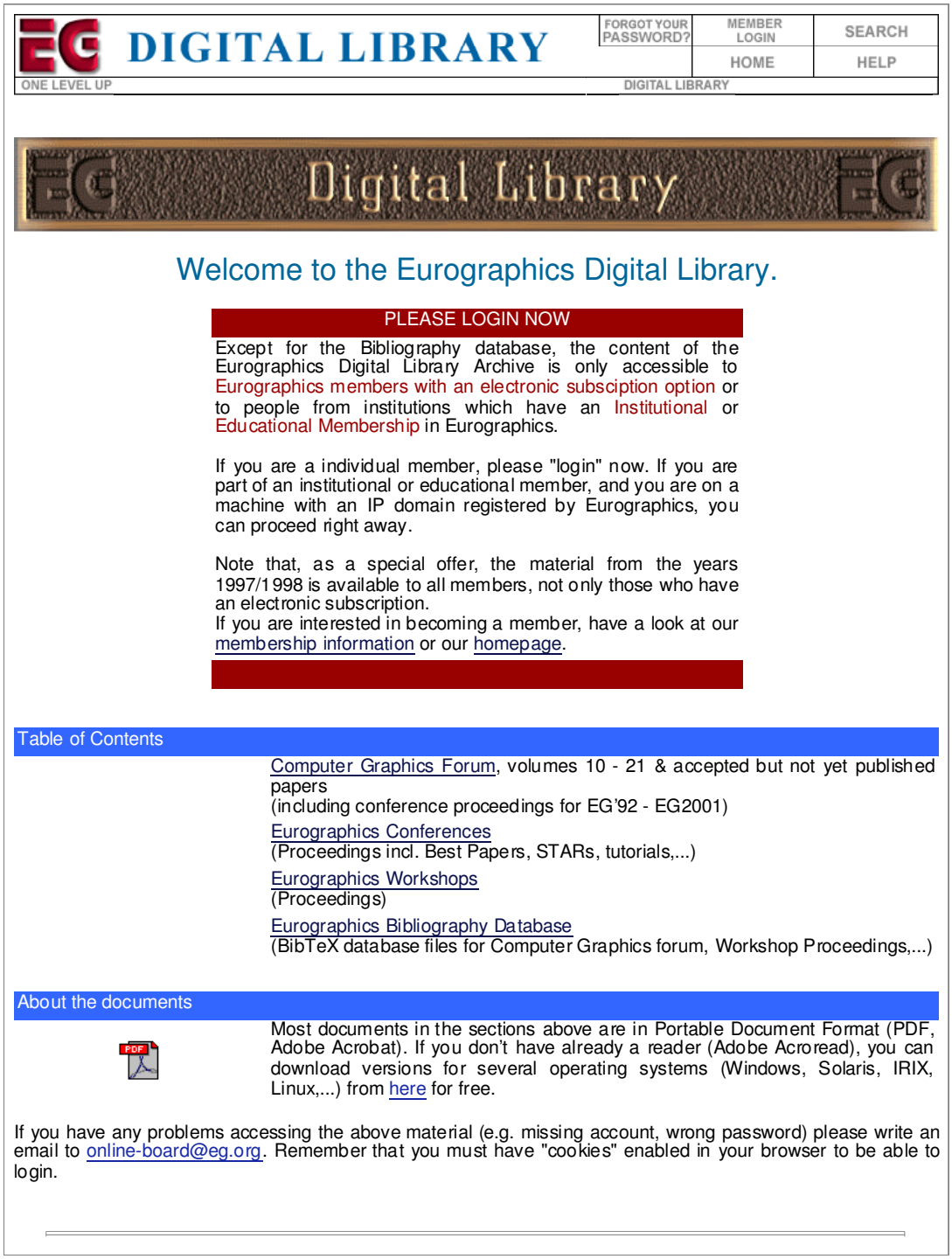


Figure 4.7: A snapshot of the *EG Digital Library* homepage at `http://diglib.eg.org`, an important part of the *EG Online* services.

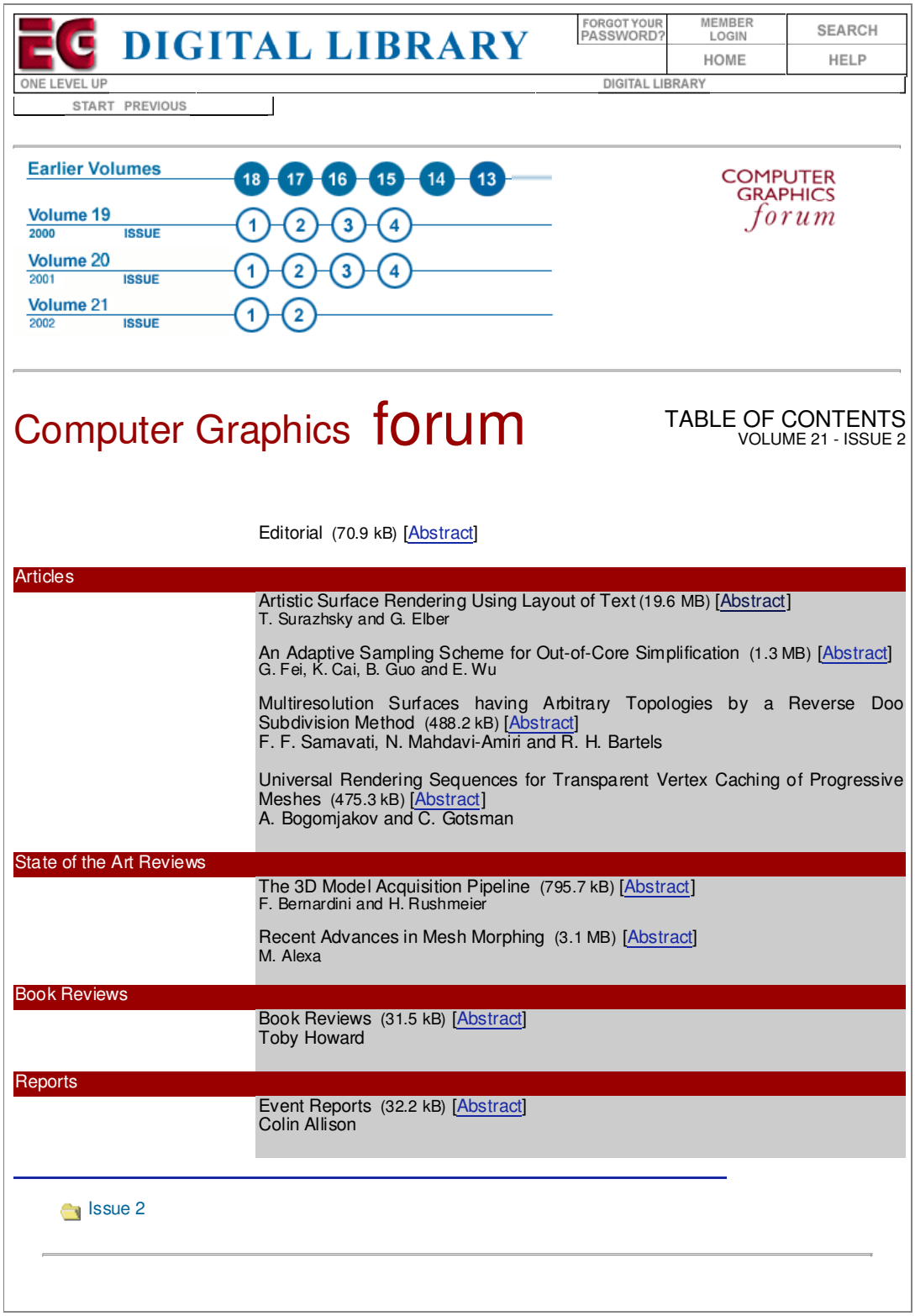


Figure 4.8: This figure shows the table of contents-page for CGF journal, Volume 21, Issue 2 in the *EG DL* as it is displayed for an anonymous user. He can only access the abstracts. See also Figure 4.9.

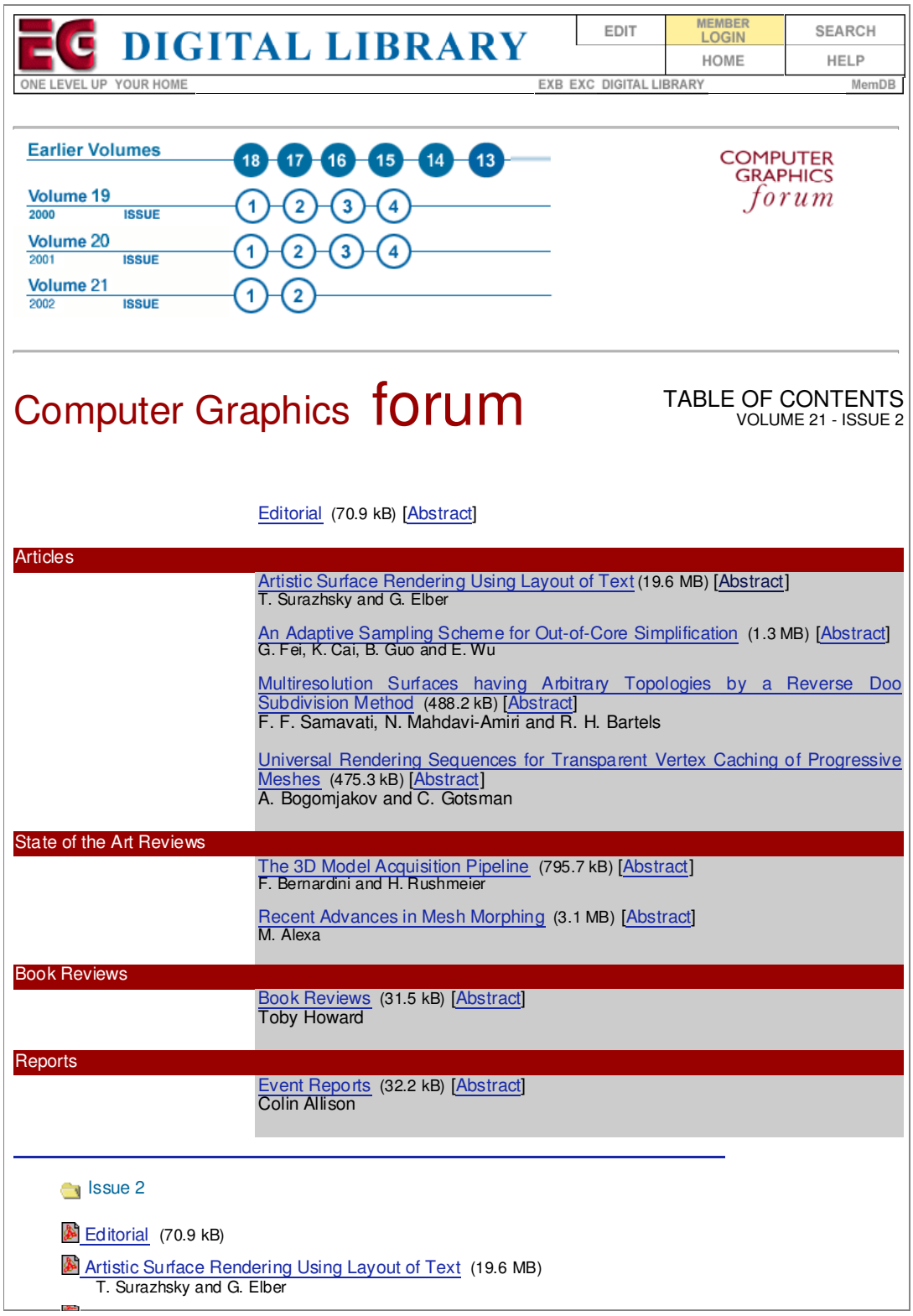


Figure 4.9: In contrast to Figure 4.8, here the user is identified and has full read access to all papers of CGF 21.2 in the *EG DL*.

mentioned, anonymous users get access to the tables of contents, to the abstracts and to award winning papers only. The same applies for members having chosen the "printed" membership option (whose number is decreasing year after year, by the way). *EG* members with electronic subscription have access to all material, identically to people in an institution with a so-called "organizational" membership (technically, everyone connecting from a group of IP numbers or with a specific hostname suffix gets access rights comparable to "electronic plus printed" subscription). There is a third group called "educational" members. This is similar to an organizational membership, but they get only access to selected materials (i.e. currently the Tutorials and the State of the Art Reports from the conferences).

The conference material section within the *EG DL* contains collections for every year (currently 1990 – 2001) with references to the conference proceedings (which from 1992 on are published as Issue 3 of the journal). All other publications (at least for the recent years) of the conferences like Short Presentations/Posters, Tutorials, State of the Art Reports etc. are also included, i.e. even publications which are never printed. The workshop section is very similar, as it contains collections with all articles and possibly multimedia documents for many *Eurographics* workshops.

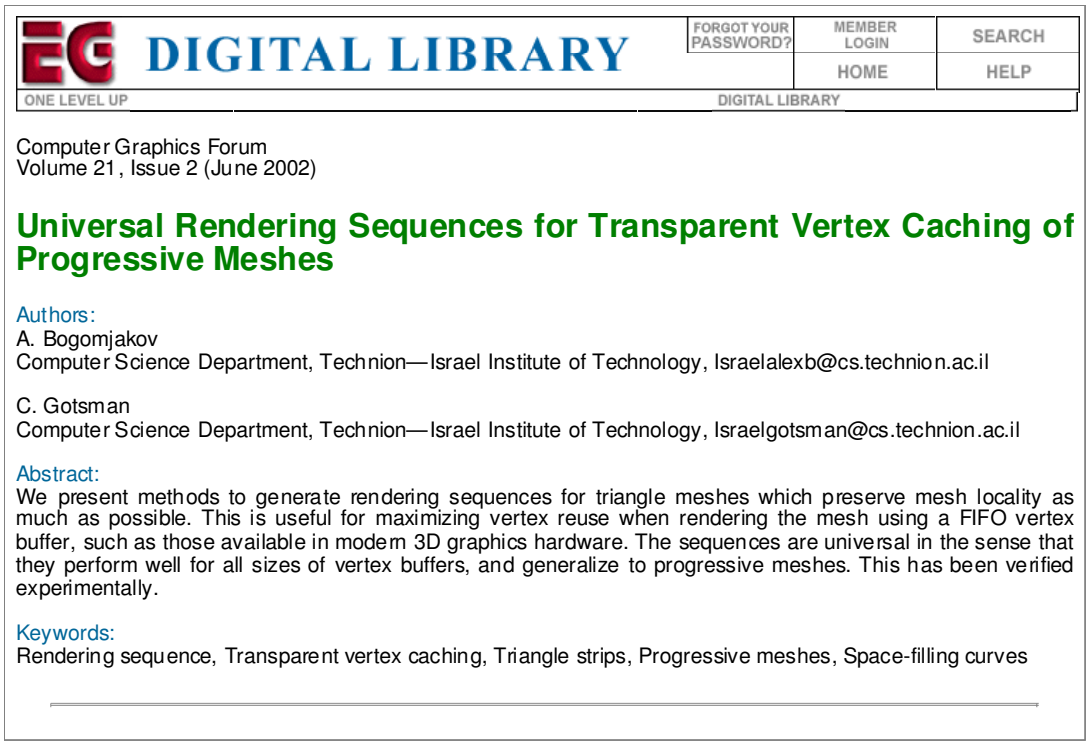
The journal as well as the conference collections form the basis for the production of several CD-ROMs (see also Chapter 5). Since a few years, the CD-ROMs of *EG* conferences are produced from the specific conference collection and the yearly journal CD-ROM is an image of the respective journal volume in the *EG DL*. To achieve this, all documents are simply dumped to a file-system, the layout and the hyperlinks are adapted by a script and the resulting file structure is burned to the CD-ROM.

The abstracts available in the journal and in the conference sections of the *EG DL* as well as other metadata are not entered manually by the administrators (i.e. us) or content providers but they are generated automatically from the metadata set prepared by the publisher (i.e. *Blackwell Publishers*, UK). This metadata is available in form of a SGML document with detailed structure. With the help of a suitable SGML parser and a simple script it is very easy to generate a HTML document containing the requested information in layouted form. This new document is then added to the *HIS* database which stores the *EG DL*. Figure 4.10 is an example of such an automatically generated abstract.

Another service offered by *Eurographics* with the help of the *EG DL* conference section is called "pre-conference access". The final versions of conference papers and articles are usually ready several weeks before the actual event because they must be printed and put on a CD-ROM (though this time-scale can be compressed significantly today using modern techniques and workflows like those discussed in this thesis, see for example Chapter 5). Now, every member of the Association is given the opportunity to access this material as soon as he or she has registered for the conference (which is usually done well in advance to benefit from a reduced fee) in contrast to everyone else who must wait until the official publication at the beginning of the conference. This is achieved by a simple technique again based on the *EG* member DB: every member having registered for the event is added to a special user group with the appropriate read rights in the *EG DL*.

The amount of materials available in the digital library is currently being increased in two ways. Of course, every new publication (journal, conference or workshop proceedings) is added as soon as the electronic version is available. Additionally, *Eurographics* has started a retro-digitization (see Section 2.2.1) project. The volumes 1 – 16 of the *Computer Graphics forum* journal were not available in digital form because authors had to deliver their camera-ready copies in print only at that time. However, the Association wants to make also these materials available to its members easily and efficiently trough the *EG Online* services. Therefore, a student project is funded where the printed versions are scanned (in full color which is very important in a field like Computer Graphics) as well as OCRed and converted to PDF documents using *Adobe Acrobat Capture*. The difference to many similar initiatives is, that not only images are stored, but the articles are really converted back to character-based text. These kind of documents then integrates nicely into the *EG DL*, e.g. the articles can be searched and found by full-text queries using the integrated search engine.

As described in the previous paragraphs above, most of the materials in the digital library of *Eu-*


DIGITAL LIBRARY

FORGOT YOUR PASSWORD? | MEMBER LOGIN | SEARCH | HOME | HELP

ONE LEVEL UP | DIGITAL LIBRARY

Computer Graphics Forum
Volume 21, Issue 2 (June 2002)

Universal Rendering Sequences for Transparent Vertex Caching of Progressive Meshes

Authors:
A. Bogomjakov
Computer Science Department, Technion—Israel Institute of Technology, Israelalexb@cs.technion.ac.il

C. Gotsman
Computer Science Department, Technion—Israel Institute of Technology, Israelgotsman@cs.technion.ac.il

Abstract:
We present methods to generate rendering sequences for triangle meshes which preserve mesh locality as much as possible. This is useful for maximizing vertex reuse when rendering the mesh using a FIFO vertex buffer, such as those available in modern 3D graphics hardware. The sequences are universal in the sense that they perform well for all sizes of vertex buffers, and generalize to progressive meshes. This has been verified experimentally.

Keywords:
Rendering sequence, Transparent vertex caching, Triangle strips, Progressive meshes, Space-filling curves

Figure 4.10: This Figure shows the abstract of a paper, generated from SGML-metadata, as it is displayed in the *EG DL*.

rographics are only available to members of the Association. A short time ago, *EG* decided that they wanted to increase the range of visibility of their products by applying a pay-per-view service. As the necessary infrastructure especially for the accounting was not available and a non-profit organization cannot afford to build up and to maintain such structures, a commercial partner was searched and found. *EG* has now signed a cooperation contract with *GetInfo*, a subsidiary of *TIB Hannover*, Germany, providing the *TIBOrder*[12] service. The new pay-per-view service which has just been established is a natural extension of the online digital library. If a non-member wants to access an article, he can find it, pay for the right to use it and finally receive it through the services of *GetInfo*. Our partner on the other hand, extracts the necessary metadata from the same SGML files which we used (see above) and downloads the actual document from the *EG DL* over a simple HTTP connection. The necessary access right is granted on an IP basis.

The final and somewhat different service of the *EG Digital Library* mentioned here is the *EG Bibliography Database*. This database is a collection of bibliographies for all publications of the Association. The bibliography data is available in form of files in BIBTEX format. This format can be used by most researchers right away, it is human readable and it can be easily indexed by all search engines (including the integrated *Verity* engine, of course). These documents are provided by volunteers from all over the world and as a consequence they are made available to everyone free of charge. Additionally we use the LATEX typesetting system to generate index files for the different information sets as PDF documents automatically.

[12] http://tiborder.gbv.de

Figure 4.11: This is the starting page of the Job Center within the *EG Online* services. See also the `SUBMIT OFFER` and `SUBMIT CV` buttons in the context sensitive menu-bar.

4.2.6 EG Job Center

The most efficient means to reach many people interested in Computer Graphics all over the world are the *Eurographics* mailing lists (see Section 4.2.4). At the time of the writing approx. 580 email addresses of members as well as 370 other recipients were listed. Several times members, colleagues and other institutions asked whether they could use this medium to distribute job offers or to seek employment. But the Association's strict mailing policy prevents this. In a time when many people receive dozens of unsolicited emails (simply called "spam") every day, one has to be very careful about what is distributed over a mailing list in order not to annoy the recipients.

On the other hand it is one of the major concerns of *EG* to help members to get their work done or to find new work to do respectively. In this situation, the idea to provide an own service for job searches and job offers (called the *EG Job Center*) was born. This new service could be easily added to *EG Online* and integrates nicely with the already established techniques and workflows. See Figure 4.11 for the starting page of the *EG JC*[13].

As can be seen in Figure 4.11, the *EG Job Center* is open to everyone and consists of two sections, a place to look for candidates and a list of jobs offered. Technically, both are hierarchies of *Hyperwave* collections which a user can browse following several different sorting criteria. The job items (i.e. either an offer or a search) are represented by special database objects containing the metadata (e.g. some characteristics of the job and a classification) and either a document or a hyperlink to a document (on the issuer's Web server) with a detailed description. Any anonymous guest can browse the complete section, but *EG* members get the advantage that they can access new items two weeks earlier, meaning a new item is not visible to unidentified users during the first 14 days. Additionally, members can set a flag in their personal data (through their homepage) if they want to be informed by email whenever a new job

[13] `http://www.eg.org/EG/JC`

offer and/or job search is added to the database.

For example, the job candidates are grouped according to country, kind of institution, qualification and kind of job whereas in the job offers section users can find collections for country, kind of institution, required qualification and type of job. The "kind of job" collection is further divided into creative, development, management, marketing, research and teaching and similar subdivision schemes are applied to the other collections. Of course, every job item is only stored once, it is then referenced from all other collections where it is assigned to (which is a feature of *HIS*).

As the *EG JC* is another Web Service, new items can certainly be added through two Web forms which can be easily reached by clicking on the appropriate links on the starting page or by using the buttons `SUBMIT OFFER` or `SUBMIT CV` in the context-sensitive menu header. The Web forms offer the possibility to upload a document or to insert a hyperlink and to type in a short description. Additionally, the item must be categorized (following the classifications mentioned above) using several predefined keywords. It is optionally possible to fill in a new keyword for every classification if the user is not satisfied with those available. Finally, the email address of a contact person has to be supplied.

For further processing, *Eurographics* has installed a "human filter", i.e. a volunteer who checks every new item before it is finally published in the *Job Center*. He or she is informed of any new item by email and can then check it, especially whether it is appropriate and whether the chosen classifications are sensible. If the item is acceptable, the so-called "JC Admin" runs a script which inserts the database object into the chosen collections, makes it visible to members and informs those members who have requested to receive such emails. If the person offering or searching a job has added new keywords (see above), the administrator has to decide whether or not to use these in the future, in which case the new collections in the hierarchy are created automatically. In the last step, the contact person of the job item is sent an email telling that the information is now online.

As mentioned, job items become visible to the public after two weeks. Also, they are removed again from the *EG JC* after six weeks. Of course, this time can be prolonged or shortened on request of the contact person. To achieve this, the JC Admin only has to run the respective script. For the other tasks, an automated script runs every night and makes items visible by modifying the access rights or removes them by deleting them from the collection hierarchy. A simple, creation time-based query to the database is sufficient to find the database objects in question.

4.3 Technical Details

In this section we will discuss some rather technical issues about the *EG Online* services (and the server) which were not mentioned so far. This will give the reader an impression about the hardware and the software required (besides the uncounted working hours of many volunteers) to run such a rather large scale installation of Web Services with satisfactory performance.

Let's start with the server hardware. Since several years now we are running *EG Online* on an ordinary but dedicated PC with *MS Windows NT4* operating system. The machine is equipped with two *Pentium III* processors running at 733 MHz, 256 MBytes of RAM and a 9 as well as an 18 GByte SCSI harddisk. It is connected to our 100 MBit in-house *Ethernet* network which has an uplink to the Gigabit backbone of the German research institutions (the *GWIN*[14]) through our University's computing center. Running a complete Document Management System (incl. WWW front-end and database) is quite a strain for this machine (compared to the resources which an ordinary WWW-server would consume), however, the performance has always been sufficient and very satisfying to both, users and administrators.

As described before, the software basis is a *Hyperwave Information Server* (see Section 3.2.2) in version 5.5 together with several *Python* [RJ99] scripts and two *Windows NT* service processes (see below), programmed in *C++*. An update to the latest version 6 is being discussed but will possibly never

[14] http://www.dfn.de/win

come, since it would require a major restructuring. For example, *HIS* 6 does not include an integrated database anymore, instead it makes use of a commercial one which has to be bought and installed separately. The layout templates for *EG Online* are based on the default templates of *HIS* 4.1. The design has been changed of course, and over the time several new functions have been added (see Section 4.2) using server-side *JavaScript* [KM97] as well as the internal *PLACE* [Hyp01] language. The *Python* scripts which are run as *CGI* scripts and the nightly "daemons" as well as the manually started administrative scripts in that programming language communicate with the server and its database using the internal TCP/IP-based protocol.

From the point on when the daily work of *Eurographics* became more and more dependent on the availability of the *EG Online* services, a big effort has been put into the reliability and into disaster recovery strategies. Of course, the internal backup feature of *Hyperwave* is run every night to copy the complete database to the central backup facility of our computing center. To reduce the downtime in case of a hardware failure or a real disaster, the Association decided to buy a second server PC with identical hardware and the same software installation to act as a mirror. This mirror server is located in another room, it is always running and connected to the network. Every night, a *Windows NT service* process runs and copies important files from the original server (i.e. the layout templates, the database, scripts...) to the backup system and also deletes files which are no more necessary. Therefore, in case of a breakdown of the *EG Online* server, the mirror contains a snapshot from at most 24 hours ago which can simply replace the primary server in a matter of minutes (only the IP address has to be changed and the *HIS* has to be started). Fortunately, until now it has never been necessary to make use of this security feature.

To further increase the reliability of *EG Online*, a second service process is running on the primary system which automatically restarts the *HIS* if it has been down for more than 5 minutes. This is useful if the server crashes (which happened frequently with earlier software versions) or if the administrator forgot to restart it after maintenance works.

Eurographics struggles hard not only to offer reliable services, but also to make these services secure. The most important example is the membership DB (see Section 4.2.1) and especially those tasks where highly sensitive personal data is involved, i.e. the application for a new membership or the renewal of an existing one. Here the user can optionally enter his credit card data to pay his membership fee easily and efficiently. Together with the other personal data this information has to be secured when transmitted over an open network like the Internet (or to be precise: the WWW). The solution applied is a transparent, encrypted and authenticated connection through the *Secure Socket Layer* (*SSL*, see for example [Tho00]). The same technique is then used to distribute the sensitive information further to the Secretary and the Treasurer. Both get an email telling that information is available and providing an URL, where they can access the data, again through an *SSL* connection. After the usage the data is deleted from the database.

SSL is implemented in most WWW browsers and in the *Hyperwave Information Server*. It secures the transmitted data by high-level encryption (i.e. in our case *RC4 128 bit*), it guarantees that the receiver is who he pretends to be by signed certificates and a chain of trust and, last but not least, it is transparent to the user: he only uses another URL starting with `https` instead of `http`.

For its online services *EG* bought the necessary certificate from a provider. This has shown to be somewhat complicated, since one of the required steps is an identity check (it is necessary to be sure that the person buying the certificate really is whom he claims to be). Usually the office of the institution to be identified is called and this is a problem with a non-profit European wide association of volunteers: there is no registered office and no full time director. Obviously the certificate provider was not prepared to handle such a situation. The solution was to specify our Institute (i.e. Computer Graphics, TU Braunschweig) as "organization" and *Eurographics* as "organizational unit". See for yourself: `https://www.eg.org`. Now, the identity could simply be checked by calling the head of our Institute on the phone.

4.4 Summary

In contrast to the application scenario described in the previous chapter, *EG Online* concentrates on Web Services as basis to access and to efficiently as well as adequately work with electronic documents and digital information on a large scale (i.e. with many users).

The first section (4.1) introduces the way from Web pages as purely static documents to a Web Service with dynamic documents, created on-the-fly, which allow the users to truly interact. *Eurographics*, the major European-wide non-profit organization for Computer Graphics, now uses a set of these services called *EG Online* for distributed collaboration and administration, anytime and from anywhere using an ordinary WWW-browser.

The Association started to use the WWW as a medium to deliver information already in 1993 and made first steps towards real services by offering an online ordering form and the possibility to upload documents (see Section 4.1.1). In 1998 *Eurographics* decided to automatize more routine work, namely the work of the Secretariat (e.g. the membership database). A low cost solution was required and found with *Hyperwave Information Server* (see also in Section 3.2.2). *EG Online* was implemented and became a success. The popularity and its usage grew, therefore it was more and more enhanced and surely the system has not yet reached the end of its evolution. The experiences are very positive from both sides, i.e. members and officers of *EG* as well as our group as implementors and administrators of the system.

EG Online presents itself in form of several Web pages with a uniform layout (called "corporate identity"), see in Section 4.2. It offers support for conferences and workshops (from simply hosting their documents up to the support of publication workflows, see *MCP* in Chapter 5), contains external sections (like for the *National Chapters* and *Working Groups*) and for the near future a new electronic voting system (*E3G*) for the election of the Association's committees is included.

The central and first component is the online *Eurographics membership database*, consisting of data about, user accounts of and homepages for the members and being used to control as well as to support most other services (e.g. by initializing access rights and updating mailing lists). See in Section 4.2.1 for a detailed discussion. The *publication section* (Section 4.2.2) is another service which was implemented very early. It consists of information about all *EG* publications, it provides guidelines and examples for authors as well as reviewers and it contains a publication catalog with a bookstore. The *document archives* as the third important service support the daily work by allowing members and officers to upload and to access digital documents, simply through a Web browser (see in Section 4.2.3). No Web Service in a strict sense are the *mailing lists*, but they are built from information in the *member database* and also hosted by our group. Therefore they do belong to *EG Online* (Section 4.2.4).

The so-called *Eurographics Digital Library* is offered as part of *EG Online* (see in Section 4.2.5). Simply speaking, the *EG DL* is an online collection of digital versions of the *EG* publications like the *Computer Graphics forum* journal or conference and workshop proceedings, together with additional multimedia attachments. As such, this Web Service is connected to the *publication section* and its fine levels of access (e.g. some documents are publicly available, others only to members of the Association with electronic subscription etc.) are made possible by the *membership database*. More than this, the *EG DL* implements several hierarchies for browsing, full-text search and a bibliography DB. The final service to be mentioned in this chapter (see Section 4.2.6) is the *Eurographics Job Center*, where job offers as well as job searches can be inserted using online forms.

The final section of the chapter on *EG Online* (i.e. 4.3) presents some interesting but rather technical issues and experiences made, like the software (i.e. *MS Windows NT*, *Hyperwave*) and the hardware (i.e. an ordinary PC) used. Reliability and disaster recovery strategies are another important point. They are achieved by using internal backup mechanisms as well as a complete mirror (based on identical hard- and software). Last but not least, security and authenticity of the services within *EG Online* is achieved by the usage of *Secure Socket Layer* connections for the transfer of and the access to sensitive data (like credit-card numbers and so on), transparently for the users.

Chapter 5

Managing Conference Proceedings

In this chapter, the background, the history, and the possible future of the *Managing Conference Proceedings* system will be presented. It was developed as part of the *WEP* project within the *Global Info* research initiative (see in Section 5.3 below). *MCP* is a system to handle and to support the submission and the reviewing of papers at scientific conferences (or similar events) as well as administrative tasks from the program committee meeting to the final pre-press steps as a complete electronic workflow. During the last months it has evolved from a prototype implementation to a complete set of powerful tools which play an important part in the publication activities and strategies of *Eurographics* (see also Chapter 4), thus being the third example of the successful use of digital documents presented in this dissertation work.

The first two sections offer an introduction to the background of the *Global Info* programme as well as to conference support software. Later, the *MCP* system is described (Section 5.4), from the basic idea to the workflow and the positive experiences made. The chapter ends with a section containing a "slideshow" of a complete example run of the system (5.5) and a summary (5.6).

5.1 Introduction

Everyone who has recently been the chairman or one of the organizers of a scientific event (be it a small workshop or even more at a large conference) has surely made one experience: the job of producing the proceedings (i.e. the final collection of the works presented at the event) is continuously getting harder, i.e. more work intensive, more complex and more difficult. Three reasons can easily be identified:

1. The events tend to grow larger whereas the deadlines for submission and publication tend to become shorter.
2. The submitted "papers" (i.e. the works to be presented) are not just textual documents anymore, often they become multimedia documents (see also Section 2.1.1), especially in fields like Computer Graphics.
3. The final proceedings of the event are no longer only a printout of the papers but often also a CD-ROM and an online presentation of the submitted materials.

The underlying workflow for the complete process of producing high-quality proceedings in printed and in electronic form is always quite similar. Figure 5.1 displays a simplified example. Typically, the whole work is centered around one person or a small group of people, usually called *Programme Committee Chair(s)* (or simply *PCC*). He or she first promotes the event (e.g. by spreading out a "Call for Papers"), receives submitted documents from the authors and then forwards these documents between authors, reviewers and vice versa. The PCC has to care about deadlines, he sends out reminders to authors as well as to reviewers and finally creates recommendations and statistics for the *PC* (short for

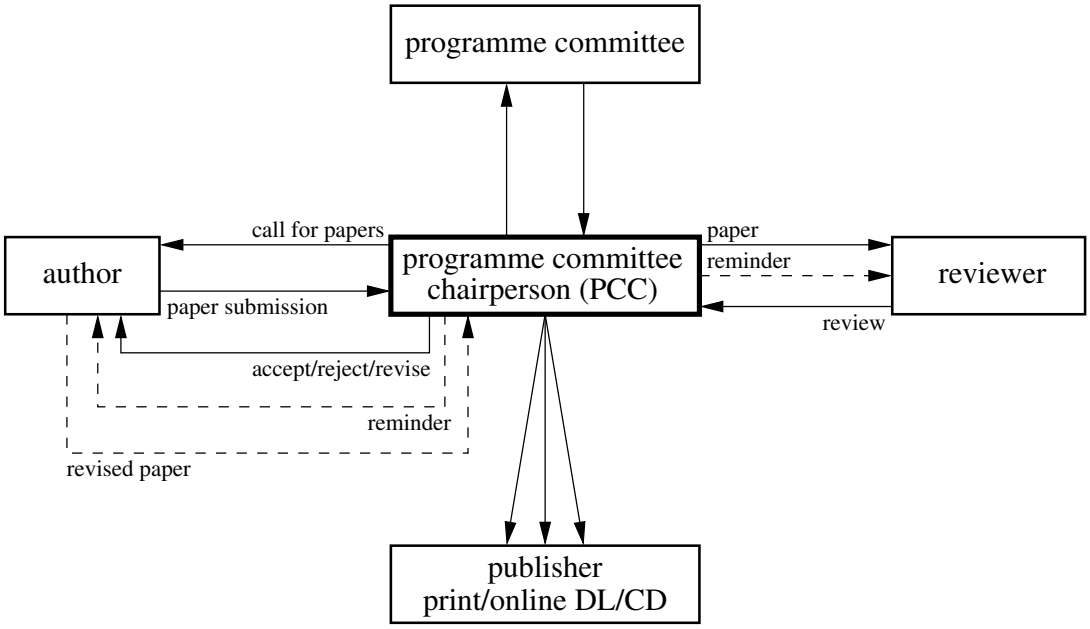


Figure 5.1: A simplified example of a typical workflow for the production process of conference proceedings at a scientific event.

Programme Committee meeting). After that, information about the decisions made (whether a submission is accepted or rejected for the event) has to be returned to the authors together with the reviewers' comments in anonymous form. As a final step, the PCC assembles one or several publications in different formats and on different media from the accepted works.

Summing up, the major tasks involved are, first, the management of the communication and the document transfer between the authors (often up to several hundred!) and the reviewers during the submission and review phase (see Figure 5.1). This part gets even more complicated if multimedia documents like video, audio or even applets are involved, the latter being very common today depending on the scientific area the conference is addressing. Second, the creation of the proceedings from the material provided by the authors is another complex task. It addresses traditional publication issues like document and multimedia formats, format conversion and metadata extraction as well as layout and presentation on different media, i.e. as a printed book, on a CD-ROM or in an online Digital Library (see Chapter 2 for a thorough discussion of these issues).

The solution for event organizers to handle these tasks can only be the use of so called *Conference Support Software*. There are already some (often commercial) tools available offering support to PCCs for one or the other of these jobs. See Section 5.2 below for a discussion. Also, publishing houses (being the traditional partners for the publication of proceedings) are beginning to use proprietary systems for their internal workflows (e.g. for the production of scientific journals – a process being very similar to the creation of conference proceedings). What we believe to have been missing until recently, however, is a system that addresses and supports the *complete* production pipeline from the author at a stage before the paper is actually written to the final proceedings, the CD-ROM and the material for the online Digital Library. A system which relieves the PCC from most of his time-consuming and technically challenging work and, thus, leaving more time to concentrate on the scientific contents and, as a result, improving the quality of the event.

Our successful answer to this demand, the *MCP* system (short for *Managing Conference Proceedings*) and the experiences we gathered during many runs at big real-life events as well as potential future improvements are presented in the following sections of this chapter, especially see Section 5.4.

5.2 Conference Support Software and Systems

As already mentioned (see Section 5.1), the difficulty and the complexity of the job to organize a scientific event and to produce its proceedings have lead to the development and the increasing usage of software systems to support these tasks. This new class of software is often called *Conference Support Software* or more general *Event Management Software* (the latter embracing many other tasks like financial calculations which won't be discussed here). Additionally, companies offering commercial support to event organizers are becoming known and used.

When we started our *MCP* project (see in Section 5.4 below) in 1998 such systems were just at the beginning of becoming available and not widely used. This has changed over the previous months and by now, many conferences are supported by software tools. But most of the systems being offered now (i.e. results of scientific projects, commercial software or simply personal tools) only implement one or several isolated steps of the complete workflow process. Therefore, they are used separately and the single results have to be combined to final publications manually in the end. What is still missing is an integrated package of tools, authoring guidelines and Groupware solutions concerted to each other, in order to support the complete publication workflow (see Figure 5.1 for a reference).

In early 1999, Rick Snodgrass has prepared a summary of Conference Management Software available at that time [Sno99] for the Executive Committee of the ACM Special Interest Group Governing Board[1]. He listed 20 different tools and created a table comparing their different features including the strengths and the weaknesses as well as the techniques used for the implementation (i.e. HTML, Perl, CGI scripting, PHP, SQL databases, ...).

In the following, we will have a closer look at some well-known systems (i.e. *START*, *WitanWeb*, *CyberChair* and the *Microsoft Conference Management Toolkit*) which are actually being used at larger events in order to give an impression of what they can and what they cannot do.

START

The original version of the *START* (for *Submission Tracking And Review Toolset*) system was developed beginning in 1998 by Rich Gerber and Jeff Hollingsworth, both professors at US universities, in order to "make the biggest drag of your life a lot easier" [Ger01] (in a way matching my conclusion from the introductory section 5.1). This early version (called *START V1*) has been used by over 550 conferences and workshops (e.g. at the *IEEE Real-Time Systems Symposiums*) and it is still available as Freeware[2], though it is no longer maintained or improved. The functionality and the features of the system are based on the authors' own experience as chairmen of scientific events and were extended during subsequent applications.

At some time, Rich Gerber resigned his professorship and now calls himself a "full-time musician and part-time software developer". He used this change in his professional career to "rethink" conference management and to analyze the "unique and complex workflow process". The result is *START V2*, a linear, process-based enterprise system to support the workflow process of running a scientific conference. This new version is now a commercial system sold by *SoftConf.com* at "flexible, but reasonable" prices.

START V2 is built to be easy to install and easy to use, therefore it is completely administrated through a Web-based graphical user interface. No difficult instructions are needed, instead all necessary information is contained on each Web page for a certain task. The system supports all jobs from the document submission to the distribution of the review results. Authors use a submission form to enter keywords and an abstract as well as to upload their paper. They can resubmit anytime and all submissions are stored in a database (sorted by title, so that double submissions can be easily deleted by the administrator), optionally converted to PDF. Finally, authors receive the automatically generated review results

[1] http://www.acm.org/sigs/sgb
[2] http://www.softconf.com

(with comments in anonymous form). Reviewers, i.e. members of the PC, have a password protected access to a personalized view of the database contents. They can access all papers online and can make their reviews via the *START* interface. Additionally, they can ask for the assignment of specific papers they are interested in and they can discuss with other reviewers through message boards. The PCC (or "manager" as he is called by the system) can watch everything and control everything. At some point he has to make the review assignments (supported by the system) and that's all. Even prototype HTML pages containing the conference program are generated automatically. See the detailed description by the author himself for more details [Ger01].

WitanWeb

The *WitanWeb* system ("Web-based Refereeing Support Software") is being developed in the Software Engineering Group at the Institute for Information Technology of the National Research Council of Canada (NRC) [Ins00] since 1997. It is well known, because it has been used for the major conference series on WWW applications, the *International World Wide Web Conference* (*WWW8 – WWW10*), amongst others. Additionally, the system is being used for the reviewing process of one of NRC's journals, so it is not limited to conferences.

WitanWeb is not only a helpful tool, but it also serves as a research platform for the development team. Research issues like Software Engineering of complex websites, Software Configuration Management, Groupware and Usability are explored in real-word applications. The system itself is based on Open Source software like the *Apache* Web server (with the *Perl* module) and the *mSQL* database as storage facility. A demonstration version[3] of the system is running and allows everyone to make a test run.

Concerning functionality and the workflow process modeled, this system is similar to *START* (see above), supporting the range from author registration to the sending of the review results, but it is not as complex or powerful. Access to the functions is granted for all parties through the WWW interface using a browser. Authors create a login account, fill out a user record (one per author), enter a paper record (one per submission) – both being stored in the database – and upload their documents (subsequent updates are possible and distinguished by a time stamp). In the end, i.e. after the PC meeting, authors receive the result including the reviewers' comments by email. The Programme Committee Chair can access all material and assigns it to PC members who in turn choose a number of reviewers. These reviewers can then login and access a personalized menu offering information about authors and submissions as well as a document download. Later, they enter their review and their comments through an online form. Based on this data, the PC members prepare a final report for the meeting. All the time, various reports matching several sorting criteria can be called by the PCC or the PC members. Special summaries to streamline the decision process as well as statistical evaluations can be generated for the PC meeting.

CyberChair

The system called *CyberChair* (or its extended, commercial version *CyberChairPRO* respectively) is somewhat different to the Conference Support Software mentioned above. The functionality and the supported workflow processes are similarly complete, ranging from the submission of abstracts to the preparation of the proceedings, but this system especially concentrates on the reviewing process as such and tries to improve it. Therefore, it is more precisely described as "web-based group decision support system to facilitate the review process" for scientific events and journals [Sta00]. *CyberChair* (also called "A Program Chair's Best Friend") has been developed by Richard van de Stadt at TRESE, the Research & Education on Software Engineering group at the University of Twente, The Netherlands, in the years 1996 – 2000. The original version[4] is still available under GNU General Public License (GPL) and, like its commercial "brother", it is constantly being used at large conferences like for example the *OOPSLA*

[3] http://witanweb.iit.nrc.ca
[4] http://www.cyberchair.org

(the *ACM Conference on Object-Oriented Programming, Systems, Languages and Applications*) and the *ICSE* (i.e. the *International Conference on Software Engineering*) series.

The strong support for the reviewing process within *CyberChair* is achieved by an implementation of several review processes identified and described in form of a pattern language by Oliver Nierstrasz in his paper *Identify the Champion* [Nie00]. Examples are named "Experts Review Papers" or "Identify the Conflicts". Other review decision patterns are used during the PC meeting like "Champions Speak First". Based on this, the system dynamically generates overviews, it contains functions to compare reviews or to point out conflicts and it offers means of communication to resolve these conflicts. Besides this special focus, *CyberChair* supports the usual tasks like the storing of author information, abstracts, papers and reviews through Web interfaces using the Internet.

By now, the author has started a University of Twente spin-off[5], where the rewritten and extended, commercial version *CyberChairPRO* has been implemented. Together with this enhanced software, the company offer their expertise, their services and their support to conference organizers, ranging from the installation, monitoring and maintenance of an installation over the conversion of document formats to the preparation or even the generation of the proceedings.

Microsoft

The final system which will be mentioned here is not very special as far as its functionality or its usage is concerned, but it has been developed by one of the major players in the global market: *Microsoft* (or *Microsoft Research* to be more precise). At least this shows, that Conference Support Software has some significance and the possible users are important enough that a *big* company cares.

The *Microsoft Conference Management Toolkit*[6] is a complete solution for the submission and review process at scientific conferences [Mic02]. It's goal is to be flexible but also easy to use, which is (again) achieved by offering all functions (including setup and configuration) through the Web browser. The software is based on *Microsoft*'s products *Internet Information Server* (for access control and GUI) as well as *SQL Server* (for storage). Besides of this, the supported workflow and the available features bear no surprises, maybe with the small exception that optionally hard copy submission can be considered. The work processes are very similar to the ones described already for the previous tools above. The *MS CMT* is now available for several years and has been used by several *EC* (for *ACM Conference on Electronic Commerce*) and *CSCW* (i.e. *ACM Conference on Computer Supported Cooperative Work*) conferences.

5.3 Global Info

In this section, the research programme *Global Info*[7] will be introduced shortly. It has set the basic framework (as far as scientific background, cooperation partners and funding are concerned) for the research project *MCP* being discussed in this chapter and especially in Section 5.4.

Global Info, for Global Electronic and Multimedial Informationsystems ("Globale Elektronische und Multimediale Informationssysteme"), has been a research programme of the German Ministry for Research and Education ("Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie – BMBF") in the years 1996 – 2000 [Glo00, NF98]. It was based on the German Government's initiative of Knowledge as Raw Material for Innovation ("Wissen als Rohstoff für Innovation") and the goal was to develop integrated scientific information systems which are to equally support all aspects of a Digital Library, i.e. from the creation of electronic scientific information and its distribution to the many different ways of searching and using it. The programme intended to foster a fundamental change of the scientific

[5] http://www.borbala.com
[6] http://cmt.research.microsoft.com
[7] http://www.global-info.org

information infrastructure and therefore all participants in the process of the distribution of information (i.e. authors, publishers, libraries, users, ...) were made to work together. Realistic developments should be initiated but basic research was explicitly not included. Instead, the major issues were all parts of the publication chain, namely:

- the annotation and the modification of content (document types; authoring tools; transfer, storage, conversion and indexing of information),
- formal description, identification, retrieval and metadata,
- the usage of content (alerting, awareness; federated systems; evaluation; autonomous agents; user interfaces) and
- economic models, billing and payment as well as statistics.

All projects dealing with one specific of these points of focus were combined to sub-programmes in form of consortia, in order to be able to closely work together and to profit from each other's results. One of these sub-programmes ("Sonderfördermaßnahmen") was named *WEP – Dokumententypen, Verfahren und Werkzeuge für elektronisches Publizieren*[8] (i.e. document types, processes and tools for electronic publishing) and our group developing *MCP* has been a member of this consortium.

The research projects cooperating in the *WEP* programme [MM-02] were concentrating on the active part of document creation, i.e. the whole publication chain. Evaluation of existing tools and workflows, standardization, optimization and improvement of tools for the production, distribution and use of digital publications were the research fields. Additionally, methods and tools for the creation of multimedia documents and the development as well as the application of working prototype systems were executed by the participating projects.

The projects within *WEP* were again divided into two groups: those concentrating on text documents and workflow and the others more oriented towards multimedia or generalized documents, together 9 projects with 14 participating institutions (research institutes, publishers and libraries). The following projects from the first group were funded (all contact addresses and more detailed information can be found on the programme's Web page, see footnote 8):

- EIT – Enabling Informal Teamwork
- Medien- und instruktionspsychologisch begründete Richtlinien für Autoren ("authors' guidelines")
- Systemunabhängige Dokumentensprachen: Textorientierte Autorentools ("text-oriented authoring tools")
- Standards, Autorenempfehlungen und Layoutvorlagen ("standards, authoring guidelines and layout templates")

The multimedia-oriented group (calling themselves *MM-WEP*) consisted of these projects (for more information, again see the URL in footnote 8):

- IPM – Individualized Print Modules
- CWB – Conversion Workbench
- MAP – Multimedia Authoring and Publishing Tools
- MCP – Managing Conference Proceedings

[8] http://vhb.fh-regensburg.de/wep

5.4 The MCP System

Similar to many of the other existing Conference Management Systems introduced in Section 5.2 above, the initial impulse to start building our management package *MCP* (short for *Managing Conference Proceedings*) [FZ01] was based on own experience. Prof. Fellner, the head of our group, had been Programme Committee Chairman (PCC) for the *Eurographics* conference (a large international conference with usually about 300 participants and 50 technical papers) in 1997, so we knew about the amount of complex work involved, although he did already try to reduce it using email and digital documents. At this time, Conference Support Software was just coming into existence and was not widely known.

Therefore, when in 1998 the sub-programmes ("Sonderfördermaßnahmen") of the *Global Info* initiative were about to start, we began developing our research project which became part of *MM-WEP* (see Section 5.3). *MCP* is intended to consider the complete publication workflow at a scientific conference, i.e. not only submission and review but also templates and guidelines for authors as well as the final publication on different media (e.g. print, online, CD-ROM). Additionally, our system is especially prepared to handle the situation at workshops and conferences of the *Eurographics Association* (see also Chapter 4): it should be freely available (which is important for a non-profit organization of volunteers), it should be adaptable (different events have different needs like for example different review criteria), it should be able to handle submissions typical for the Computer Graphics field (i.e. containing much multimedia material) and it has to be fully usable through the WWW (as authors, reviewers and PC members are distributed all around the world). Also the development in close contact with the actual users was important to us. Because we were seeking immediate feedback and comments, each prototype has been used to handle one large scientific conference as a real-life test.

The basic framework as well as the implementation details of *MCP* (see in Section 5.4.1) were derived from experiences gained in our previous (or at that time still ongoing) projects *GoldenGate* (see Chapter 3) and *EG Online* (Chapter 4). The resulting system has permanently been used at the *Eurographics* conferences and at several workshops where it has proven to work very successful. It is described in detail in the Sections 5.4.2 – 5.4.4 below and an example run is presented using many screenshots in Section 5.5. In Section 5.4.5 we will discuss the future possibilities of *MCP* considering extensions to the system as well as an increased usage.

5.4.1 Basic Idea

From our own experience and from the experiences of others, we are convinced that the only way to substantially reduce the workload for programme committee chairs of a scientific event is the application of a complete and electronic document managing and publishing process. Such an optimized workflow is based on the following four aspects:

1. **digital multimedia documents**
   All "papers" are created, submitted, stored and reviewed in electronic form, in a well defined format, with any type of multimedia element (i.e. video, audio, ...) embedded or attached.

2. **automation**
   Routine work is done automatically, only under the control of the PCC but not by himself. Less trivial tasks are at least made significantly easier and less time-consuming. For example, submissions are to be received and forwarded with the least necessary interaction and reminders on various deadlines are issued automatically depending on each individual's previous actions.

3. **tools**
   Every actor in the publication process is provided with a set of "tools" supporting all necessary actions most efficiently. This includes word processor templates and layout instructions for authors as well as tools for the PCC to forward the submitted documents to the reviewers and to return

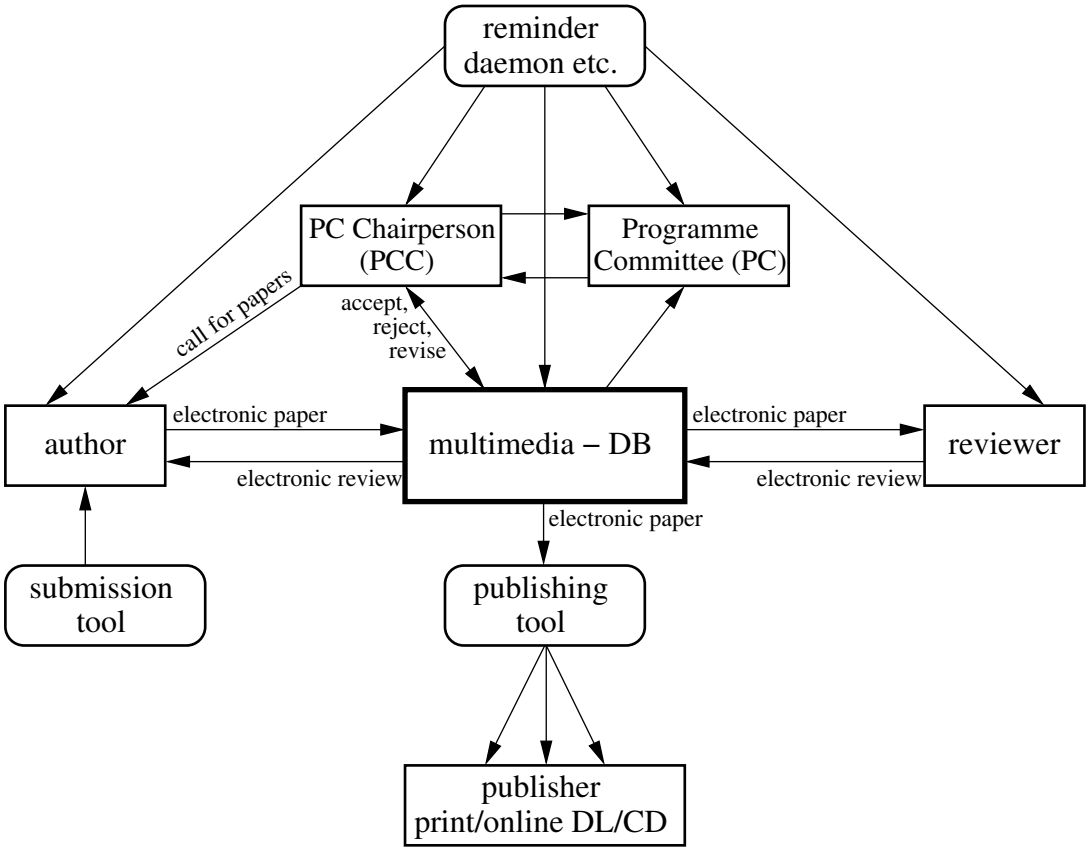


Figure 5.2: An example for a complete electronic workflow for the production of conference proceedings as implemented my *MCP*. See also the traditional paper-based workflow in Figure 5.1.

certain portions of their comments to the authors. Last but not least, the final version of each submission has to be converted to the formats necessary for the different publications.

4. **Web Services**
   The tools, features and functions mentioned in the first three aspects above all have to be offered through WWW interfaces (in form of Web Services), so that they can be used or accessed through ordinary Web browsers anytime from anywhere without having to install special software.

In summary, an efficient and complete workflow will support the input (i.e. formats, templates, data mining), the managing (i.e. submission, review) and the output (i.e. cross-media publishing) of "papers" as well as other, secondary tasks (e.g. reminders, email handling) with the help of Web Services. A workflow for a scientific conference which exploits all these aspects will be similar to the one schematically displayed in Figure 5.2.

In contrast to the usual (non-digital) workflow as presented in Figure 5.1 at the beginning of this chapter, a multimedia database now becomes the central component and the main workhorse. As a result, the PCC now only has to do a small remaining set of tasks on his own:

- The PCC sets up and initializes the system (i.e. the database, the tools and the daemons to automatize secondary tasks).
- The PCC sends out a *Call for Papers*.

- The authors create their multimedia documents with the help of a *submission toolkit* (e.g. layout templates and authoring guidelines) and submit them directly to the database electronically.
- The submissions are forwarded to the reviewers automatically and electronically. Of course, the assignment of papers to reviewers is controlled by the PCC or the PC members respectively.
- All parties involved are kept up-to-date and they can contact each other by email. Also deadlines etc. are announced through this communication channel (e.g. by a *reminder daemon*).
- The reviewers return their results to the database, again directly and electronically.
- After the PC meeting, submissions are marked in the database as accepted or rejected.
- Authors are informed automatically. This includes the sending of reviewers' comments intended for the authors in anonymous form. Later, revised documents are received in a way similarly to the original submission.
- Finally, a *publishing tool* (e.g. layout templates, format conversion scripts, ...) extracts the accepted documents from the database and starts the different publishing processes (in the formats agreed upon).

The *Managing Conference System* implements this new workflow with the help of techniques which we explored and developed in previous projects presented in this work (e.g. software components and multimedia Document Management Systems, see Chapter 3, or Web Services, see Chapter 4). See a detailed discussion and how this approach has proven to be very successful in the following sections.

5.4.2 Workflow Elements

Our *Managing Conference Proceedings* system implements the complete electronic workflow as given in Figure 5.2. For the central multimedia database we used a *Hyperwave Information Server* (see [KMS93, Mau96] or in Section 3.2.2) which is a Document Management System that can be simply described as the blending of an object-oriented multimedia database and a WWW server. It offers database-like features such as indexing (including full-text indexing for a large variety of document formats) and hierarchical structures as well as bidirectional persistent hyperlinks, a fine grained access control and layout templates, all through a standard HTML-interface. An API for direct TCP/IP-communication with the document management engine is also available. As we have shown earlier, a system like this can be easily used to implement a simple form of Workflow Management (see [FZ99b] and in the previous chapters, especially Section 3.2) which makes it a good choice for a Conference Support System.

The active components of *MCP* such as user interaction, the "daemons" and automated emailing (like event notifications) are either implemented by layout templates of *Hyperwave* (using client- and server-side *JavaScript* [KM97]) or by scripts using *Python* [RJ99], which can be run as CGI-scripts as well as stand-alone. Simple emails (like the acknowledgment of an upload) are generated on the fly, while more important ones (like the review results) are prepared by the organizers, containing some keywords which are then automatically replaced. Another important role is played by the document templates for the authors (currently LaTeX 2ε and *MS Word*) and the pre-press toolkit (see in Section 5.4.3 below) which together allow to generate final versions, suitably formatted and prepared for the different publishing processes. In this section I will now concentrate on the single elements of the workflow; more technical details considering *Hyperwave*, the implementation of systems by combining standard components and Web Services can be found in the previous chapters 3 and 4.

Figure 5.3 gives an overview of the implemented workflow subdivided into four subsequent phases (i.e. *installation*, *submission*, *review* and *publication*). As illustrated, each actor in the process (*administrator*, *author* and *reviewer*) is associated to one or several tasks. Now, first a short overview of the different phases will be given, before they will be discussed in more detail later in this section.

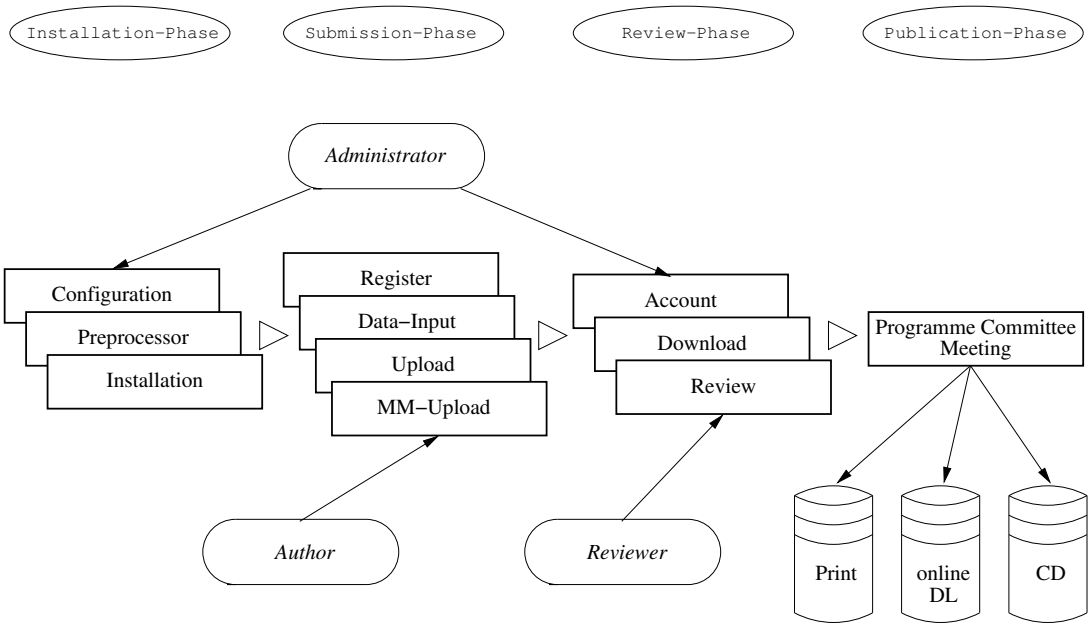


Figure 5.3: This figure shows the workflow process, its elements and the participating actors, as implemented by the *MCP* system.

At first, in the *installation phase*, the *administrator* (represented by one or several persons in real-life, for example – but not necessarily – the PCC) configures and installs the system. This includes specifying the name of the event and the publication, entering deadlines, setting hostnames or email addresses and so on.

After that, the so called *submission phase* begins during which the authors play an active role. As a first step, each author has to register with the system. Entering name and email address allows all further communication to be email based. On the server, an account (i.e. username and password) is created which allows the author to create a new submission (entering requested data and uploading his "paper") or to access one of his previous ones. At any time, authors can download the document templates and examples which are provided by the system.

Following the submission deadline, the *review phase* is initiated by the *administrator*. Essentially, the following happens: the access rights for the submissions are changed in a way that authors have no longer write access and the reviewers' accounts are created. A list of *reviewers* (with name and email address) and a mapping between reviewers and submitted papers is provided by the Programme Committee Chair. The PCC can access all submissions with their metadata as well as several statistics to make this assignment. A reviewer can access and download exactly those submissions assigned to him and has two options to deliver the reviews: either by filling out an online HTML form or by first downloading a text document (in XML format), adding the review offline and returning it by email to the *MCP* system where it is parsed and stored automatically.

When the reviewing is finished and the Programme Committee meeting has decided upon which submissions to accept, the *publication phase* begins. Now, authors are informed of the review results by automated emails and they can provide final versions of their work in a way similar to the first submission. From those the final publications (e.g. a combined PDF with page-numbering etc.) can be prepared using our pre-press tools. Additionally, the available metadata can be used to build an online version and/or a CD-ROM.

In the following we will discuss these four phases in more detail.

Installation Phase

In this first phase, the *MCP* system is set up, installed and configured. Possibly by the Programme Committee Chair himself, by a secretary, by a student assistant or even by a conference managing company (further down called *Administrator* after the role in the workflow process). The only prerequisite is the installation of an *Hyperwave Information Server* (at least version 4.1) and a *Python* interpreter on the computer acting as the server. The actual system consists then of a set of scripts, several templates both for Web pages and for emails as well as a configuration file.

At first, this configuration file (being a simple text-file) should be modified according to the *Administrator's* needs. Generally this means doing some global settings (like: what is the name of the event? what are the hostnames of the server itself and of the SMTP server used for sending emails? etc.), specifying information about the papers to be submitted (e.g. how many pages are allowed or whether the authors have to enter an abstract first) and choosing one of several options according to the submission and the review process (like the deadlines) which will be explained later in the appropriate phases. At this time, also the HTML and email templates can be modified as needed.

Next, a template preprocessor script is run which uses the information given in the configuration file to finalize the layout templates, the Web pages and the emails (the latter being sent automatically by the system, e.g. when an author registers, see below). After that, if the installation is done remotely i.e. not from the server itself, the *Administrator* can copy the scripts and the configuration to the server machine. As a last step, an installation script is run which sets up *Hyperwave* and its internal database (inserting the Web pages and CGI-scripts). Now, the server has to be restarted and the show can begin.

In the configuration file several other persons can be specified (with name and email addresses) who get full access to all materials of the system and who can execute many administrative functions (like manually adding a new author or a make a submission), called the *Publication Admins*. They can also receive (if requested by setting an option in the configuration) system messages by email, whenever an author registers or a new document is uploaded etc. By this function and by the ability to read all collections on the server as well as to call dynamically generated statistics (e.g. see Figures 5.11 and 5.18 in Section 5.5 below) the *Administrator* and the *Publication Admins* are always up-to-date.

Submission Phase

During installation, *MCP* prepares a "submission information" Web page which acts as a homepage for authors when the *submission phase* has begun. This page describes how the submission process works ("You have to register first..."), what the requirements for submitted documents are and it gives other information that the PCC might have added. This page also contains links to the document templates and examples which are included.

In order to be able to upload submissions, authors first have to register with the system. This is easily done by following a link from the homepage and entering name as well as email address into a Web form (see Fig. 5.6). Following, a unique account (i.e. a username/password combination) is generated for the author (and sent to him by email, Fig. 5.7) which on one hand enables him to actually make one or several submissions or to access these again later and on the other hand guarantees that the author can be identified and contacted by email.

Once registered, an author can access the submission page using another link from the homepage and logging in to his account. Here, the user can click on two hyperlinks. The first link brings him to a list of all his previous submissions to this event (see Figure 5.13 for an example) while the second one creates a new submission or technically speaking, a submission collection, as a submission consists of several data- and document objects (containing metadata about the authors and the contribution as well as the actual paper and possible multimedia attachments). As a result, an email containing the unambiguous identifier (i.e. "<publication><number>" like "paper42") and the direct URL of the new submission is sent by *MCP*. Generally, all major actions of the author like registering, submitting or uploading are

confirmed by an email. Also the *Administrator* (and the *Publication Admins*, if requested) receives short system information emails. Additionally, the actions as well as their results or possible errors are written to log-files in these cases.

A submission collection (be it newly created or visited again later on) is presented to the user in form of a Web page listing all the information entered and all files uploaded so far. By clicking on the appropriate links more data can be added, existing information can be modified and documents can be freshly uploaded as well as updated, all through specific Web forms (see Figures 5.8, 5.9 and 5.10 in Section 5.5). In order to ensure that at least the metadata required for a submission to be valid is available, the respective fields have to be filled out sequentially before optional information can be given. This means for example that first a primary author has to be entered (which has not to be identical to the person having done the registration, of course) and then one or several secondary authors can be added. The email addresses given here are later used as recipient addresses for all messages concerning this submission (e.g. the review results).

Before the actual document can be uploaded, some metadata describing this contribution has to be entered (like the title or the number of pages). The *Administrator* can request (by setting the appropriate option in the configuration file) that also an abstract is to be entered here. Another option is that authors have to supply a fax containing a copyright statement (being prepared by the system) at this stage. In a final step, i.e. after the "paper" itself has been added, additional multimedia documents can be uploaded. All uploaded files must be in one of the predefined formats (usually PostScript or PDF for the actual paper, preferably compressed archives for the multimedia attachments). Again, an author can always come back to continue where he left or to update previous uploads and to change data already specified (only before the submission deadline, of course).

All the actions described above, can also be executed by the *Administrator* (and the *Publication Admins*) at any time, using the so-called "Admin Menu" (a part of it is displayed in Figure 5.12). This is a Web page which contains many compact Web forms for the creation of a submission, for the upload of a document etc. These are of use for example when an author is unable to use the WWW-interface because he has no or only an insufficient Internet access. Additionally, from the "Admin Menu" all submissions (and later on all reviews) can be accessed as well as the "Statistics Menu". This is another Web page only readable by the administrators which offers to display several dynamically generated statistics (e.g. see Figures 5.11 and 5.18). Lists of registered authors, of complete submissions, of available reviews and so on can be generated in both human- and machine-readable formats by simply clicking on a link. Finally, information about all submissions or the submitted documents themselves can always be "dumped" into files by calling one of the Python scripts in the *MCP* installation package.

As an automatic control mechanism, all newly uploaded compressed archives (`.zip` and `.gz`) are checked for completeness and data integrity nightly (i.e. by a "daemon" script run every night by the *Hyperwave* server). If an error is found, the *Administrator* is informed by an email. He can then contact the author to request a corrected version. Additionally, the state of all documents (`unchecked`, `ok` or `corrupt`) is indicated on the Web page of each submission (see in Figure 5.10).

Another feature has been added to the latest version of *MCP*: papers can already during the *submission phase* be marked (see the respective function in Figure 5.12) as to be reviewed by the *Next Year's Chairs* (*NYCs*). This means, that the *Publication Admins* will no longer be responsible for the reviewing of these submissions, even more, they will not know the assigned reviewers nor will they get to see their results (at least not within the system). Instead, someone else will take care of this, which is a useful function if some of the organizers want to submit own papers to the event. For the *Eurographics* conferences, where *MCP* is used, this job is taken by the chairmen of the conference in the following year (the *NYCs*). As soon as the responsible mark is set, they are informed by email (being inserted into the configuration file), they can access the submission from a special list and so they can start to look for suitable reviewers.

Review Phase

Once the submission deadline has passed, the *review phase* is initiated by the *Administrator*. He can start the reviewing for all submitted papers at once or optionally for some of them only. This can be done by starting a script or through the already mentioned "Admin Menu". What happens is that, first, the access rights for the submissions are changed so that the authors can no longer write or modify them (they retain read access, though). Second and more technically, the submission collections are moved to another location within the *Hyperwave* database hierarchy so that they now are found in the list of papers being reviewed (which is also reflected by changed numbers in the statistics). And finally, a submission is presented differently in the Web browser: it is no longer a form but a simple display of the information. Additionally, the assigned reviewers and their results are presented (to users with proper access rights, of course, see Figure 5.16), if available.

As an optional feature, all members of the *Programme Committee* can get access to the complete set of submissions, thereby providing a significantly higher level of transparency for their work. The *Administrator* only has to provide a list of name/email-pairs to the system which will then create the required accounts and inform the *PC* members by email about their access.

What now has to be done is the assignment of the submitted papers to the available reviewers. Therefore a list of reviewers (with name and email address) and of the papers assigned to each (by simple listing the unambiguous IDs) has to be prepared. This can be done by the *Administrator* and the *Publication Admins* or the *PC* members, with the help of several displays, lists and statistics which the system offers. For the appropriately marked papers (see above), the *NYCs* play this role, of course.

Here it makes a difference whether a one-stage or a two-stage reviewing approach is chosen (i.e. it was configured during the *installation phase*) which are both supported by *MCP*. Two-stage reviewing means, that there are two kinds of reviewers, namely *Primary* and *Secondary Reviewers* doing two separated rounds of reviewing. A person can belong to both groups, of course. During the first round (called *Review Phase I*), every reviewer (i.e. *Secondary* and *Primary*) makes a normal review of the submissions assigned to him and at that time he or she can only access the paper itself as well as his or her own review. When the *Administrator* switches a submission to *Review Phase II* (usually when all submissions from the first phase are available), all *Primary Reviewers* are asked to read the reviews from round number one (which they now can access) and to write a summary review which can then be presented at the *PC* meeting to help to speed up the decision. The *Primary Reviewers* are informed about the change of state for each single paper assigned to them by email. In contrast to this, one-stage reviewing consists only of *Review Phase I* and only *Secondary Reviewers* exist (who are thus simply called *Reviewers*).

Another option which can be requested by setting a flag in the configuration file is *blind reviewing*. This means that the reviewers will not get to know the origin of the papers they are reading. Both authors and reviewers are informed of this on the respective Web pages. The authors are asked explicitly to not include their names and organizations into the actual documents, however, they must enter this information during the *submission phase*. But when *blind reviewing* is active, the access rights to these data objects are modified as such that reviewers cannot read this hidden information.

During the reviewing itself, a reviewer logs in using his account, after which he is given a Web page containing instructions and a link to a list. This list contains exactly these submissions which were assigned to him (see Figure 5.13). From here, he can read all the metadata of the submissions (besides the exception of *blind reviewing*, see above), he can download the papers and all multimedia attachments and finally provide his evaluation and his comments (i.e. the review). A reviewer has usually two options to deliver the reviews: either by filling out an online HTML form (see a part of it in Figure 5.14) or by first downloading a simple text document (in XML format, see Figure 5.15), adding the review offline and returning it by email to the *MCP* system where it is parsed automatically. In both cases, the data is stored in database objects, with the effect that the current review results can always be displayed in several formats on-the-fly and statistical overviews can be generated dynamically. In any case, the receipt of a new review is acknowledged by email and it can be overwritten or modified at any time later on (before

the respective deadline).

The *Administrator*, the *Publication Admins*, optionally the *PC* members and optionally the *Primary Reviewers* (in *Review Phase II*) can access all available reviews in different formats (i.e. HTML, XML, CSV, e.g. see in Figure 5.17) through the Web page of each submission or through the functions of the "Statistics Menu" at any time (see Fig. 5.18). Additionally, the "Admin Menu" offers simple functions to create new reviewer accounts or to reassign and to unassign reviewers at any stage. Other scripts allow to send emails to all reviewers who have not yet provided their comments (e.g. when the deadline approaches), to contact *Primary* and/or *Secondary Reviewers* or to dump all available reviews to a file-system in HTML/PDF (e.g. to be comfortably presented at the *PC* meeting).

Publication Phase

When it has been decided which submissions to accept for the event (usually by the *PC* meeting), these results (i.e. a boolean value like `accept` or `reject`, others are possible e.g. when a revision is necessary) have to be added to the system's database, thus initiating the *publication phase*. A simple text file matching submissions IDs with results is sufficient. When *MCP* processes this file, it adds the data to each submission collection (meaning it is also displayed by the appropriate Web page and used by the dynamically generated statistics). Additionally it sends an email to each author. This email is based on a template prepared during the *installation phase* (see above). The results and the comments of each review available for the paper in question are added automatically and, of course, in anonymous form.

Further emails to all authors of accepted contributions can be easily sent by the *Administrator*. Usually this is used to inform the authors on how to prepare and how to provide the final version ("camera-ready copy") for publication. This final step can also be supported by *MCP* if requested. In this case, all accepted submissions are switched to a so-called *Final Submission Phase* which is very similar to the original *submission phase*. Authors have now again write access to their submission collection (using the already known accounts and URLs) and can upload the final version of the paper as well as multimedia attachments, using nearly the same Web forms and techniques as previously. Some events alternatively ask the authors to upload their documents to a FTP server or simply send them by email.

When all documents for the proceedings are collected (be it PDF documents or multimedia files) they can be dumped to a file system by running a script. After that, they could be handed over to traditional publication channels. A common presentation is guaranteed, when the authors are made to use the provided layout templates. As an alternative, the available metadata could be used to build online proceedings and CD-ROMs using for example stylesheets (like *XSL* see Section 2.1.5) or simple scripts, as we did for the CD-ROMs of several *Eurographics* conferences and the *CGF* journal's yearly CD-ROMs (see in 4.2.5). For a printed publication most of the necessary pre-press steps can be executed using a toolkit included within the *Managing Conference Proceedings* system (see in Section 5.4.3 below). The final result would be a single PDF file which can be sent directly to the printer.

5.4.3 Pre-Press Production

Once all the final documents building the proceedings of an event or the issue of a journal have been collected, they will be published through several channels (e.g. online, on a CD-ROM or in printed form). Usually, after some proof-reading, the files for the printed version are simply forwarded to a publishing house. Additionally, a running order is defined, a preface and maybe an index are added etc. by the *Programme Committee Chair* (to use the example and the terminology of the previous sections). Pre-press steps like the final assembly and the "polishing" are left to the experts who take their time and often also a substantial amount of money (either directly or by selling the proceedings).

Beginning with the availability of new publication techniques (like printing-on-demand or online publication) and the everyday usage of digital documents (e.g. in PDF) it is now possible to create professional looking publications on your own, i.e. without the help of a publishing house. This has

advantages both in production time as well as in production cost and often is the only feasible solution for a non-profit organization like a scientific community of volunteers. During such a pre-press production process many problems have to be solved, though. Several of these are listed and further discussed in the following paragraphs:

- The individual "papers" have to be assembled to a single book, a proceedings volume or a journal issue, i.e. into a single and complete new document which can then be sent directly to a printer. This is more than just joining pages, since also a specific running order has to be considered and often the first page of an article needs extra treatment (see below).
- A consistent page-numbering has to be added (which could start with any value, like for the third issue of a journal in a year), taking care of different physical positions for odd and even pages as well as for different page layouts and sizes.
- Often, though a unique page layout was applied, the documents received from the authors contain pages of different sizes (like A4 or US Letter etc.) and/or with different content offsets. To build the assembled document these have to be normalized and aligned to a uniform raster.
- As a final step it might be required to add copyright information or a "brand-mark" (e.g. to indicate the origin of a document) to the first page of the proceedings or each single article respectively. This is especially useful for documents which are to be published in digital form.

For the *MCP* system and all the events (from the field of Computer Graphics as well as Digital Libraries) where it has been used by now, the format of choice for the final documents is PDF because it is well-defined, it is platform-independent, reader applications are freely available and it supports an exact and professional looking layout (see in Section 2.1.3). PDF has another advantage: the necessary pre-press steps listed above can all be performed by manual interaction using the commercial *Adobe Acrobat* package, which offers functions to join and reorder pages, to add page content and to change the location of document elements.

However, depending on the size of the planned publication (i.e. the number of single documents to be assembled) this can be a very tedious and error-prone work. Therefore it is essential to have a set of tools which automatize these tasks as much as possible. Since the *Management Conference Proceedings* system has been designed to support the complete workflow of the production of publications it does contain such a package, called the *CGF Toolkit* (because it was first used for an issue of the *Computer Graphics forum* journal).

The *CGF Toolkit* is a set of currently five plug-ins for *Adobe Acrobat* or programs using the *Acrobat SDK* respectively, which means that the software from *Adobe* has to be installed to use them. The following are available:

`CGFpagesizer`: to set all pages of one or a group of PDF documents to the same size, defined by the user (see Fig. 5.4, on the left),

`CGFnormalize`: to adjust the upper left edge of each document page's content to a given offset, being identical for all documents to be assembled (left half of Fig. 5.4),

`CGFpaginate`: to add consecutive page numbers to all pages of a document at the correct physical page position, depending on whether an odd or an even page is treated,

`CGFbranding`: to insert a copyright information or a "brand-mark" on each first page of a set of PDF documents (see Fig. 5.5), and

`CGFmerge`: to combine several PDF files into one new document while inserting empty pages if necessary, to have each original document begin on an odd page (see also on the right of Fig. 5.4).

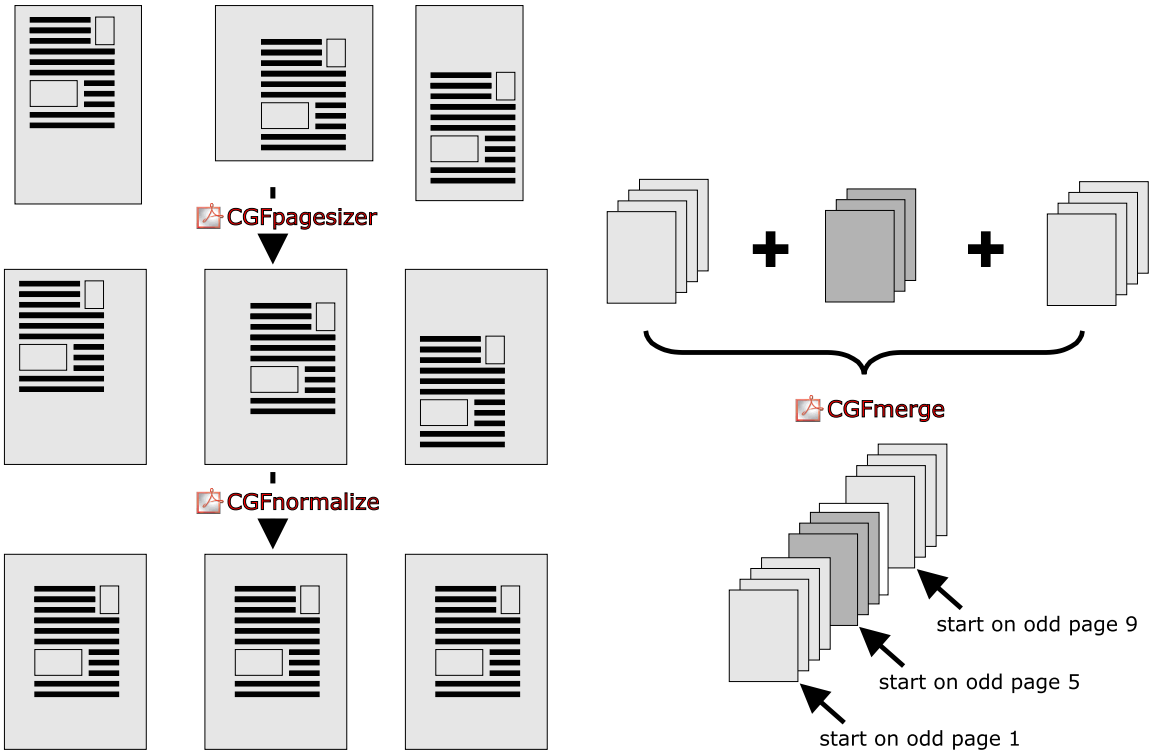


Figure 5.4: Two diagrams schematically showing the results of the `CGFpagesizer`, `CGFnormalize` and `CGFmerge` tools being applied to PDF documents.

Figure 5.5: A screenshot of a a “brand-mark” added to a PDF document by the `CGFbranding` tool.

All tools are operated by choosing the appropriate menu item within *Acrobat*, except for `CGFmerge` which is a stand-alone application. When started, each tool opens a dialog box to enter offsets or to choose options and possibly to select a group of files. After a job has been started, a progress bar indicates the current state while one or several modified PDF documents are being created.

From the technical point of view, all tools but `CGFmerge` are implemented as plug-ins using the SDK for *Adobe Acrobat 5*. `CGFmerge` on the other hand is a *MS Visual Basic* application accessing the functionality of *Acrobat* via the so-called *Interapplication Communication* (*IAC*) of *Adobe* [Ado01].

`CGFpagesizer` simply sets the `CropBox` and the `MediaBox` (defining the visible area and the physical size respectively) of each page in a PDF document to given values. The next tool, called `CGFnormalize`, then calculates for each page the bounding box of the complete content, computes its distance to a given offset and finally moves every content element by this vector. See also the diagram in Figure 5.4. The vertical position on every page to place the page number at is retrieved from the page's bounding box by `CGFpaginate` while the horizontal position has to be specified by the user (separately for odd and even pages). The `CGFmerge` tool assembles several documents to a new one, in a specified order. If one of the documents to be merged has an odd number of pages, an empty page will be added after the last page, so that each individual "paper" starts on an odd page (see in Figure 5.4). Additionally, the first page of each original document can be extracted to a new file, e.g. to serve as an abstract. `CGFbranding` finally, simply adds a logo (which can also contain hyperlinks or all other page elements supported by PDF) to the first page of one or several PDF files at an arbitrary position (see an example in Figure 5.5). In one step, it also changes to the document security settings can be applied (enabling or disabling the ability to print or modify a PDF document). In a typical workflow (like the ones described in this chapter) these tools would be run in the order in which they just have been described.

The *CGF Toolkit* has been used successfully to combine and to prepare the printed as well as the online proceedings of the *Eurographics 2002* conference. For our experiences made, see also below in Section 5.4.4.

5.4.4 Applications, Advances & Experiences

The *Management Conference Proceedings* system described in this section has by now been used successfully at three *Eurographics* conferences (see also in Chapter 4) and several Workshops. Additionally, some of the tools included (like for example the *CGF Toolkit*, see Section 5.4.3 above) were used to produce CD-ROMs and documents in the *Eurographics Digital Library* (see 4.2.5). The following list gives an overview about these reference applications, ordered by date.

***Computer Graphics forum* journal, volume 18 (1999):** production of annual CD-ROM

***Eurographics 2000* conference (EG2000):** submission and review of Full Papers and Short Presentations (2 separate workflows); production of conference CD-ROM

***Computer Graphics forum* journal, volume 19 (2000):** production of annual CD-ROM

***Eurographics 2001* conference (EG2001):** submission and review of Full Papers and Short Presentations (2 separate workflows); production of conference CD-ROM

***Int. Symposium on Virtual Reality, Archaeology and Cultural Heritage* (VAST2001):** full paper submission and review

***ECDL Workshop on Generalized Documents* (2001):** submission, review and assembly of final paper versions; production of printed proceedings and CD-ROM

***Computer Graphics forum* journal, volume 20 (2001):** production of annual CD-ROM

***Eighth Eurographics Workshop on Virtual Environments* (2002):** paper submission and review

	submitted	*total amount*	*smallest*	*largest*	*MM ratio*
EG2000 – Full Papers	138	480 MB	42 KB	56 MB	73 %
EG2000 – Shorts	35	97 MB	44 KB	29 MB	84 %
EG2001 – Full Papers	174	960 MB	68 KB	75 MB	75 %
EG2002 – Full Papers	232	1202 MB	257 KB	411 MB	65 %

Table 5.1: This tables presents some interesting numbers about the papers being submitted to several *Eurographics* conferences and thus managed by the *MCP* system. For each event, the number of submissions made, the total amount of data (partly compressed), the sizes of the smallest and the largest submission as well as the ratio of multimedia material in the complete set of submissions is given.

***13th Eurographics Workshop on Rendering* (2002):** pre-press steps during production of proceedings (printed/online/CD-ROM)

***Eurographics 2002* conference (EG2002):** submission, review and final assembly of Full Papers and Short Presentations (2 separate workflows); production of printed proceedings (incl. all final pre-press steps), see [DS02]; production of conference CD-ROM

***GRAPHITE 2003* conference:** submission and review of Papers, Short Papers, Digital Art Gallery as well as Electronic Theatre submissions (4 separate workflows); currently ongoing

Now, some detailed experiences made especially during the major events (i.e. the *Eurographics* conferences) will be presented in order to illustrate the problems we encountered and how the system has been improved as well as extended to avoid them.

An early prototype of the *MCP* system has been used to manage the complete submission and review process of Full Papers and Short Presentations at the *Eurographics 2000* conference. This application has to be considered an elaborated field test. A second version, improved and extended as a consequence from experiences during the first run has then been used in the following years (see below).

This stream of conferences (being the second largest scientific conference on Computer Graphics world-wide) has been regularly attended by more than 300 participants over the past years. About 50 scientific papers are presented, together with other material like *Short Presentations*, *State-of-the-Art Reports*, industrial presentations and tutorials. The proceedings are published as one issue of the *Computer Graphics forum* journal as well as in the *Eurographics Digital Library* and on CD-ROM. Table 5.1 gives some numbers on the amount and sizes of the documents submitted and thus handled by *MCP*. As can be seen, the number of submissions to deal with as well as their size has been steadily increasing. Also the ratio of additional multimedia material is very high (although it was lower at the last conference which might be a sign that authors start to *embed* high-quality images into their documents). This is not surprising for a Computer Graphics event, of course, but it also shows that authors accept the feature to provide attachments and make use of it. Generally, the amount of data to be handled is significant, therefore these applications were a real stress test for *MCP*.

Already during the first run in the year 2000 we gathered a lot of results, most of these have been confirmed in 2001 and 2002. Additionally, we gained some interesting insight into authors' behavior which we would like to share. First of all (and probably most important), right from the beginning the system worked well, even beyond our own expectations. The PCCs of all three events confirmed that they "just had to wait until everything was ready" and thus they were very satisfied with the *MCP* system, which can also be read in the prefaces of the printed proceedings [GH00, CR01, DS02]. Additionally, we made the following observations and as a conclusion made the modifications or extensions mentioned:

FTP vs. HTTP:

During the first run (i.e. the Full Papers submission at *EG2000*) many authors had problems with the HTTP-upload of large files (see the maximal sizes in Table 5.1). Conventional browsers do not give any visual feedback (e.g. a progress-bar) and the receiving process at the server is not started before the transfer is completed. Therefore, the browser seemed to be "frozen" for a long period of time (and sometimes really was). As a result, many users aborted the upload (leaving incomplete files in the DB), retried it several times (creating many incomplete submissions) or asked for help.

As an experiment, we then allowed the upload by FTP for the Short Presentations. This worked better from the user's perspective but it is less comfortable, i.e. more manual and difficult steps are involved. Also a lot of control is lost on the server side: it is not possible to manage what is uploaded and it is difficult to identify uploaded material automatically ("Which filename belongs to which submission?"). Another solution would be to use a *Java Applet* (see for example [AGH00]) for the upload process, which could then provide visual information about the progress. We did not follow this road because it would be against our philosophy that special software should not be necessary at the client. Additionally, some users have disabled *Java* and for our applet to work, it would have to be granted file-system read-access which is a security risk and thus declined by many users.

For the second major run of *MCP* at the *Eurographics 2001* conference, we again used HTTP-upload, but we allowed and encouraged the authors to send several files one after another, e.g. splitting large archives in parts and especially separating the actual document from its multimedia attachments. This time, we achieved much better results. Only a very small number of upload problems were reported although authors really did stress the system (e.g. for *EG2002* a complete CD-ROM of about 411 MB was successfully uploaded). Other reasons for the better performance surely are that on one hand people are now more used to this kind of Web Service and that on the other hand the network infrastructure is constantly improving (i.e. offering a faster throughput). As a conclusion, the simple HTTP-upload through a Web form definitely is the way to go.

network problems:

Often the network itself was the source of problems. The Internet and its technologies are still not as stable as we all like them to be. Additionally, more and more restrictions are put onto corporate network access or Intranets. As a result, authors were not able to connect, *Firewalls* prohibited the upload of files (necessary for the document submission), browser *cookies* were filtered out (avoiding successful identification of authors or reviewers) etc.

When offering a Web Service, situations like these always have to be taken into account, i.e. alternative means of communication (like FTP or Email) and a support hotline have to be installed. It is also necessary to exactly log what is going on at the server, to be able to identify whether the problem is on the server or on the client side or whether the user reporting the problem simply made an error (see below). Additionally, as experience shows, the time is on our side because users tend to get more and more used to situations like these and also how to avoid the possible pitfalls.

user errors:

One well-known though sometimes hard to learn lesson is that it is never possible to predict completely what a user will do or is trying to do. Certainly, this statement is always valid but it should be especially taken into account when designing and running a Web Service which will possibly also be used by computer illiterates. The proper thing to do is to give the user some freedom of choice when to do what, to allow him to change his mind and to enable him to modify previous actions taken. On the other hand it is necessary to offer him as much assistance as possible and sometimes even gently push him in order to try to avoid that something goes wrong.

This was one of the bigger extensions we implemented for the second version of the *MCP* system. During the first run, we made the author insert the metadata and upload his documents in one go, after which he could never get back to his submission. As a result, we received many requests for a change of address, for a revision of the uploaded document etc. that we all had to care for manually (which btw. was simple through the WWW-interface but however, it had to be done). Therefore, the second version allowed the authors to perform their steps arbitrarily during several sessions. They can now always come back to change some information or to revise their uploaded document (see Section 5.4.2). The latter has proven to be in great demand, e.g. for some submissions we received more than a dozen revisions, sometimes in a time-frame of minutes.

missing functionality:

Once a new Conference Support Software is installed and users (i.e. the *PCCs*, *administrators*, *authors* and *reviewers*) get used to it, they will exploit its functionality. Soon they will come up with new and sensible ideas. One example is the possibility to update the metadata and to revise the documents, requested by the authors (see above), something they would never have done (at least not to this extent) if the classical paper-based approach had been applied. Another example are the statistics and the evaluation functions for the Programme Committee Chairs. Questions like "Can the system give me a list of *all* authors?" or demands like "I want to send an email to these reviewers who have not yet provided all their comments." have contributed to the increasing number of functions which each new version of the *Managing Conference Proceedings* system offers.

As a conclusion to these experiences, a system to manage conference submissions and reviews has to be flexible and extensible. This is a requirement which makes the applied technology (i.e. a multimedia Document Management System, WWW interfaces and script programming or more generally speaking, the combination of standard components and Web Services, as developed in chapters 3 and 4) especially appropriate (see also Section 3.2.3).

PostScript vs. PDF:

The discussion about the "best" document format concerning complete electronic publication workflows is still ongoing, different parties have different preferences (e.g. presentation-oriented, structured, proprietary or open standards). See Chapter 2 and especially Section 2.1 for a detailed discussion. For scientific publications an exact layout is often required and images or diagrams need to be embedded, hyperlinks for references are also helpful. Last but least, it must be easily possible for the authors to create documents in the chosen format. As the most often used authors' tools fulfilling these requirements today are *MS Word* and (in some fields) LaTeX [Lam85] the selection is effectively reduced to formats which can be written by these programs (set aside whether this is good or bad, again, see Chapter 2). This is one of the reasons why *MCP* includes templates for exactly these tools. Unfortunately, the native format of *MS Word* (`.doc`) is proprietary and platform-dependent while the `.dvi` of LaTeX lacks the ability to embed fonts or images. If one then goes one step further, the solution is PostScript [Ado98] or PDF [BC93]. Both can be created from the source easily (by using *Adobe Acrobat Distiller*, the *GhostScript* Freeware or simply printing to a PostScript printer driver and storing the result).

By definition, *MCP* does not care about the format because it receives and stores document files (of any kind). It is possible though, to require special formats (special filename extensions, to be precise) during upload. For all events supported by the system so far, the authors were required to upload a PDF or PostScript file (as the actual "paper"). To some of them, it still seems to be a problem to generate such files without errors. The only solution is to come up with detailed descriptions on how to proceed, what to do and what not to do (although such information is often ignored) and to wait for the development of improved software.

It should be stated that most of the time, PDFs received from the authors work without problems, but

some errors are repeated again and again. For example, it happens that special fonts (often containing symbols or Asian characters) are not embedded. The resulting problems range from a bad formatting when printed to the impossibility to open the document at all for reading (i.e. when a PDF containing references to a non-embedded font with Asian encoding is opened by *Adobe Acrobat Reader* without the Asian font packages[9]). Sometimes, a PDF document is very large and one specific image on a page is displayed very slowly (literally pixel by pixel). When trying to print this document it takes even longer or the printer simply aborts the job completely. This happens, when an image containing "transparent" pixels (e.g. because of an *alphachannel*) is embedded. As PDF does not support images with transparencies, these are reproduced by representing each pixel of the image by one rectangle in the appropriate color. This results in many thousands or even millions of objects on one page (e.g. an A4-sized image at 300 dpi contains approx. 8.7 million pixels), which often is too much for the memory, the software or the patience of the reader. The authors have to be asked to deliver updates in these cases.

The latest extension to the publication workflow supported by *MCP* was the usage of the pre-press tools described in Section 5.4.3 for the printed proceedings of the *Eurographics 2002* conference [DS02]. During earlier events, authors sent their final documents either electronically to the Programme Committee Chairs who prepared the publication or directly to the publisher as printed Camera Ready Copies (CRC). For *EG2002* we received PDF documents from the authors and performed all final pre-press steps on our own using the *CGF Toolkit*, meaning we prepared *one* PDF file which was directly sent to the printer.

This worked quickly and smoothly. The only problems which occurred were due to malformed PDF documents, the wrong layout or too many objects on a page (see above). For one of these pages we had to sent a revised version to the printer, because they were not able to handle it though we could. Because of the short time-frame we were not able to get back to the author with this. The solution was "quick and dirty": we displayed the problematic image with maximal screen resolution and took a screenshot. The old image was then removed (with a text editor in an extracted PostScript file, as *Adobe Acrobat* could not handle the page) and replaced by the new one. Of course, in the future such images have to be avoided right from the beginning.

The final PDF to be sent to the printer was about 100 MB in size, containing 406 pages. It had the correct page-numbering, the page sizes were right and all page content was aligned equally. The proceedings printed from this file show a good quality and can not be distinguished from issues of the previous years as far as the look & feel is concerned. Therefore, also this final step in our workflow can be considered as a big success, one which will surely be repeated (see in Section 5.4.5 below).

5.4.5 The Future

The *Managing Conference Proceedings* system has come already a long way during the previous years. The support of the workflow at the *Eurographics 2002* conference marks an important milestone because *MCP* has now reached a state which we call a "production release", meaning the software is now ready for the daily use, it is complete. Of course, a good software always evolves and is never really finished, but it can now be applied by anyone who knows how to handle a computer, email and PDF documents. He or she can use the software successfully (as we have proven many times, see above) to support the submission, the review and the publication process at a scientific event and that was the initial goal (see Section 5.1).

However, possible and useful extensions to the present version are already clearly visible. It is only necessary to cast a look at other Conference Support Systems, for example those listed in Section 5.2. Instead of implementing one specific step, *MCP* supports the complete workflow from the authors to the final product. As a result, some of the intermediate tasks definitely could be implemented better or smoother. Examples are the review assignment and the automatic generation of indexes or structures for

[9] http://www.adobe.com/products/acrobat/acrrasianfontpack.html

an easier publication (or even the preliminary program). A database of reviewers (belonging to an event series, of course) could be integrated which contains also preferences of the reviewers and their present research activities. Based on this knowledge and keywords or a classification by the authors, proposals for good review assignments could be made automatically. Other systems allow reviewers to have a look at the submitted papers and "choose" papers they would like to review, which is another option.

The support for the electronic publication process could be extended in a way so that an online presentation or a CD-ROM is generated automatically. Possibly XML (e.g. XHTML [W3C00a]) files would be created from the available metadata and a default stylesheet for the layout (e.g. XSLT [W3C01c]) would be provided (see also Section 2.1.2). Finally, the user only needed to adapt this to his personal needs. A third point is flexibility. For example, currently the system includes one specific review form in an offline and an online version querying several quality measures (with some options). What if an event organizer would like to emphasize other items? Just think of another preprocessor which prepares all review forms, regarding the interests of the PCC. The opportunities are endless and the problems challenging...

The other interesting question regarding the future of our *Managing Conference Proceedings* system is how it will be used and by whom. There is currently a trend in many research societies, organizations and libraries to start taking the (especially digital) publication and the distribution of scientific material in their own hands. There is a lot of political and economical reasoning involved, which won't be discussed here. Those more interested could start off with the final chapters of [FE00] or with many publications from the latest conferences and workshops about Digital Libraries (like this year's *6th European Bielefeld Colloquium*[10]) and from the *Open Archives Initiative*[11] (like [RF01]). Another entry point could be the revolutionary (some they "populistic") theses of biochemist Pat Brown from Stanford who postulates to give scientific articles away for free and supports this through his *Public Library of Science*[12].

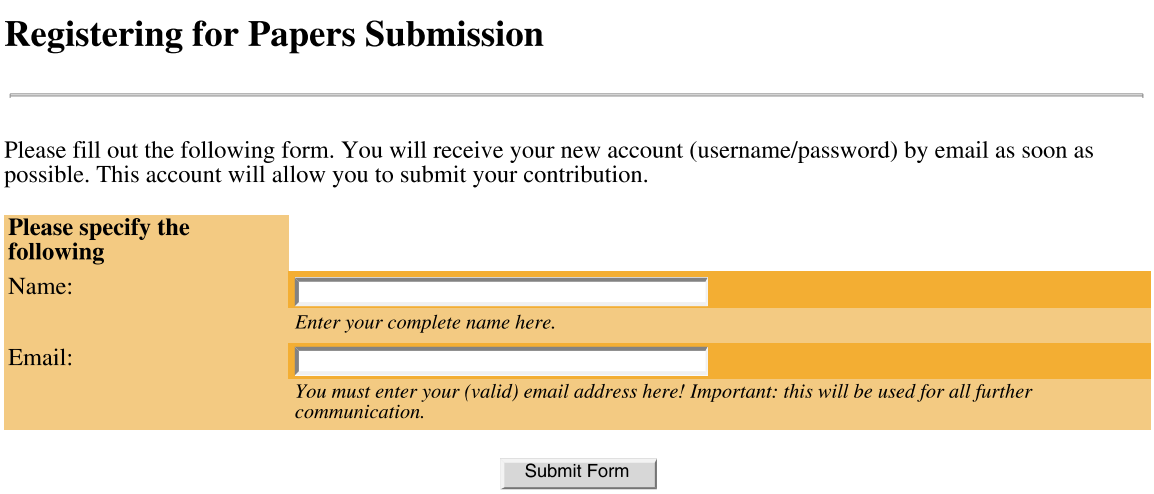
To come back to my point, the most important asset for a publisher of scientific information is content, high quality content to be precise. In many cases scientific organizations own this intellectual property already (by organizing events or by running a journal) and they are responsible for the quality control (usually by peer-reviewing). So the answer to the question how for example *Eurographics* (see at the beginning of Chapter 4) gets the content when they want to become a "publisher" is very simple: exactly as they did before. The important question is rather how to collect the content, how to manage it, how to execute quality control and how to achieve a cross-media publication efficiently, quickly, with the least possible amount of work and at a low cost. Which brings us back to Event Support Software like *MCP*.

Recently, the *Eurographics Association* decided to buy the support through the *MCP* system for its conference and for five workshops each year (together with the *EG Online* services and the *Eurographics Digital Library*, see Sections 4.2 and 4.2.5) as one step on its way to becoming a publisher for the owned scientific information. So, the further and increased application of *MCP* on a nearly commercial and regular basis is now guaranteed. The idea is not only to "sell" a software system but also a suitable workflow process (incl. example emails and document templates) and expertise gained.

5.5 Complete Example Run

During the last sections, Conference Support Software (Section 5.2) was described, especially including our *Managing Conference Proceedings* (Section 5.4) system as a concrete implementation. The sequential workflow elements as well as the basic ideas have been discussed in detail. *MCP* has been used to support the publication process of proceedings at several big events during the last three years (see

[10] http://www.ub.uni-bielefeld.de/2002conf
[11] http://www.openarchives.org
[12] http://www.publiclibraryofscience.org



Registering for Papers Submission

Please fill out the following form. You will receive your new account (username/password) by email as soon as possible. This account will allow you to submit your contribution.

Please specify the following

Name:

Enter your complete name here.

Email:

You must enter your (valid) email address here! Important: this will be used for all further communication.

Submit Form

Figure 5.6: Using the Web form displayed in this figure, every author has to register to the system.

```
From: eg02admin@cg.cs.tu-bs.de
To: m.zens@tu-bs.de

Hello Marco Zens!

Thank you for registering as author for EG02.

You have been assigned the following account to submit your Full Paper:

  username: eg02paper_1001
  password: 1ZT5D

--
(This email has been generated automatically.)
```

Figure 5.7: This figure shows an example of the email received by an author upon registering for an event supported by *MCP*.

Section 5.4.4). Now, screenshots and documents from these applications will be used to illustrate a complete example run, i.e. in a way "replay" the process as described by Figure 5.3 on page 97 in form of a "slideshow" from the points of view of all participating actors.

Imagine the Programme Committee Chair of an international, scientific event has set up an instance of *MCP* (see Section 5.4.2 for a more detailed description of each workflow step), he has distributed a Call for Papers and now an author wants to make a submission. He (or she respectively) has found his way to the Web pages and has read the instructions carefully.

The first step is to register for the event in order to get an author account. This is done simply by filling out a Web form like the one in Figure 5.6, which results in the receipt of an email containing the information necessary to login to the system (see in Figure 5.7, as an example for the many automated emails sent by the system).

After the author has logged in successfully, he creates his new submission and enters the required information. This is done again through Web forms. Any data entered is displayed on the Web page of this submission and can be extended or modified later on. The screenshot in Figure 5.8 shows what has

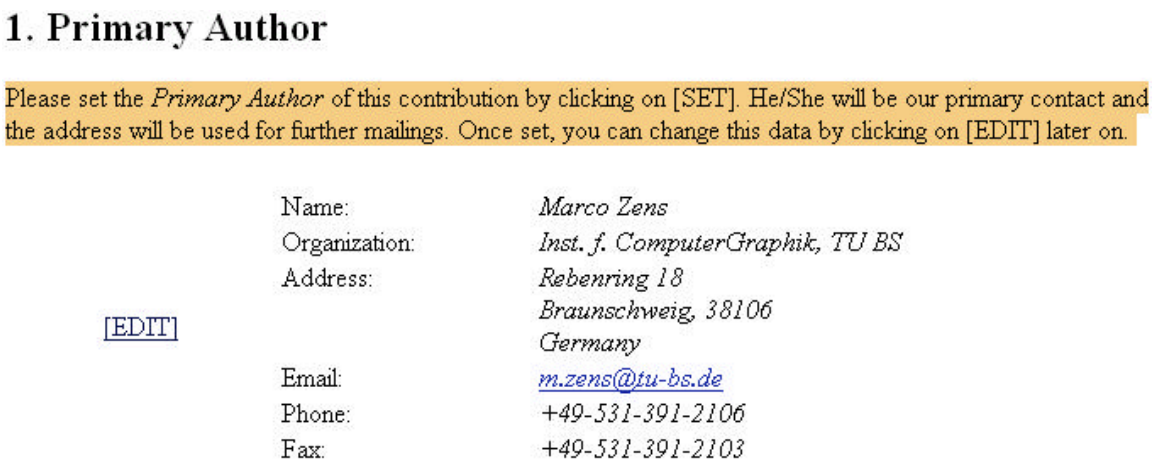


Figure 5.8: This screenshot shows an author's contact information being displayed in the Web browser when accessing the submission collection.

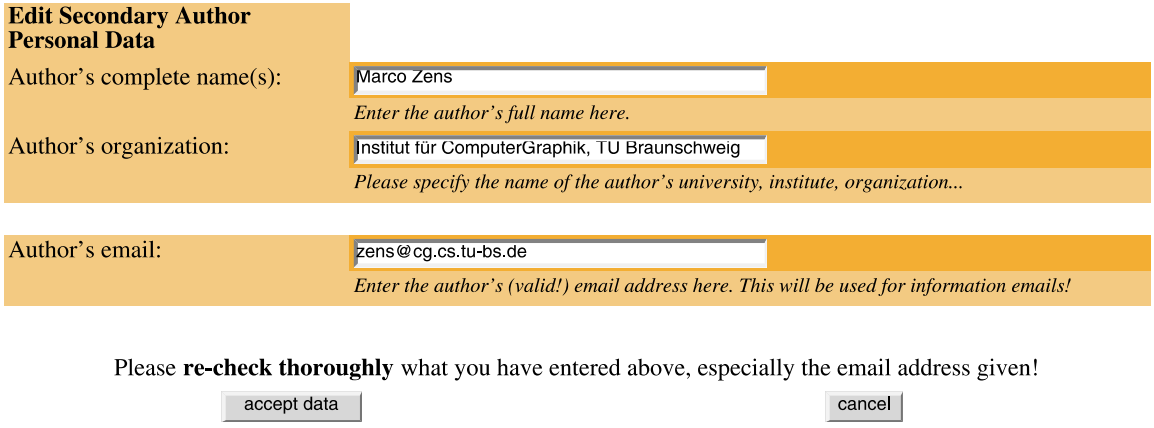
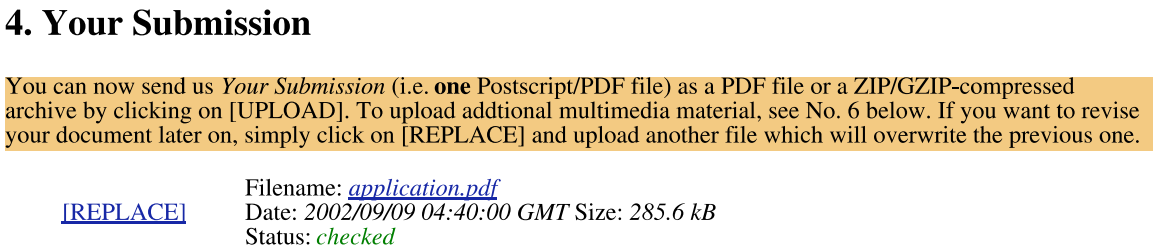


Figure 5.9: Here in this figure it can be seen how the personal data of a Secondary Author is entered/modified.

been entered under the Primary Author section. It could be modified by clicking on `EDIT` which would reload the Web form again, this time pre-filled with the data already available. The next figure (i.e. 5.9) is another example for such a Web form to enter data. Here a Secondary Author is added by giving the name, the affiliation and his email address.

The most important step is of course the upload of the actual contribution to be submitted. This is done through a third form which only requires the local filename to be entered. When the upload has been successful, this is acknowledged by an email and displayed on the submission's page, see Figure 5.10. In this special case the filename (i.e. `application.pdf`), a time stamp and the size of the document are shown. The status `checked` is an indication, that the file has been checked for consistency during the nightly run of the appropriate daemon (see Section 5.4.2). The author could click on `REPLACE` anytime (i.e. before the deadline) to revise his submission by uploading an updated document.

The Administrator and optionally the PCC (who could be the same person, of course) are informed of creating- and writing-actions (like preparing a new submission or uploading a new document) by emails. They can also use the online statistics which are dynamically generated on-the-fly by choosing a link from the "Statistics Menu" like the "Current Submission Status" displayed by the screenshot in Figure



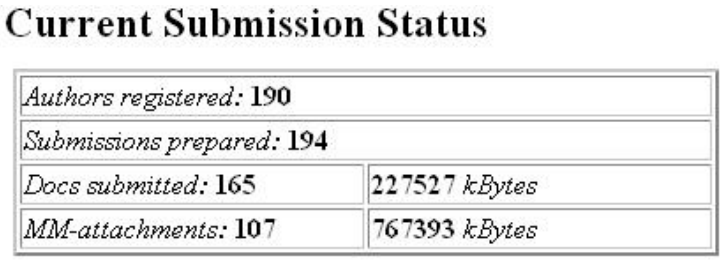
4. Your Submission

You can now send us *Your Submission* (i.e. **one** Postscript/PDF file) as a PDF file or a ZIP/GZIP-compressed archive by clicking on [UPLOAD]. To upload addtional multimedia material, see No. 6 below. If you want to revise your document later on, simply click on [REPLACE] and upload another file which will overwrite the previous one.

[REPLACE]

Filename: *application.pdf*
Date: *2002/09/09 04:40:00 GMT* Size: *285.6 kB*
Status: *checked*

Figure 5.10: The part of the submission form where the actual submission can be uploaded or revised (as in this special case).



Current Submission Status

Authors registered: **190**	
Submissions prepared: **194**	
Docs submitted: **165**	**227527** *kBytes*
MM-attachments: **107**	**767393** *kBytes*

Figure 5.11: This screenshot shows one of the many available, dynamically generated statistics. In this case, a simple overview of the current submission status is displayed.

5.11. As an example, here by 190 registered authors, 165 contributions were currently submitted (a total of $\approx$ 227 MByte). Obviously a higher number of submissions (i.e. 194) have been prepared already, so there is more to come.

While the process is running, the Administrator has also the opportunity to intervene, i.e. to take own actions. This could be either to support users or to prepare the reviewing process. Most of these administrative actions can be performed through the Web-interface. Figure 5.12 shows a part of the "Admin Menu", the latter being the full collection of available functions. Using the GUI elements of the small section presented in the figure, an invalid or incomplete submission can be deleted, a submission can be assigned to the Next Year's Chairs (see under *Submission Phase* in Section 5.4.2), a submission can be moved to the next phase (i.e. the *review phase*) and finally a reviewer can be assigned to a submission.

Back to the author: he can access any of his previous submissions by choosing from the list presenting exactly those that are accessible. See the screenshot in Figure 5.13 for an example. It shows a fake submission which I did during *Eurographics 2001* (and which certainly has been deleted shortly afterwards). Similar lists are available for the PCC and optionally the other Programme Committee Members (containing all submissions) as well as the reviewers (containing exactly the submissions assigned to them) when the reviewing has begun.

Some time after the submission deadline, all complete and valid submissions are switched to the *review phase* (see the appropriate paragraph in Section 5.4.2 above). Now the reviewers can access them, download the documents and multimedia attachments and prepare their evaluation. The actual review is then provided either by filling out an online Web form (see Figure 5.14 showing a part of it) or an offline text form (i.e. an XML document, see Figure 5.15 for a shortened example).

The reviewer retrieves the offline document from the Web page with his instructions and the HTML form by clicking on the `ADD` hyperlink in the list of reviewers of each submission (similar to Figure 5.16). The part of an example online review form in Figure 5.14 contains an interesting detail: behind `Authors` it reads `(Blind Reviewing)`, meaning that the blind reviewing option has been chosen for this event. Therefore, the reviewer will not get to know the authors of the paper he is reviewing (if

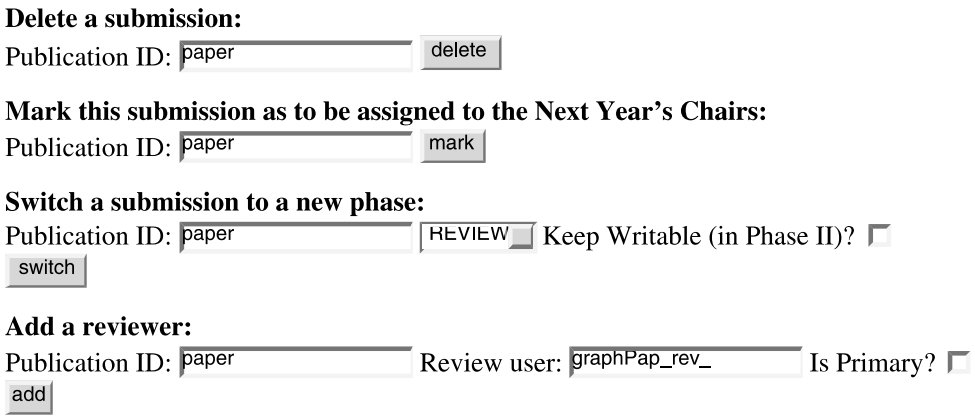


Figure 5.12: A small part of the "Admin Menu" is displayed in this screenshot. The Administrator only needs to fill out some fields and click a button in order to delete a submission etc.

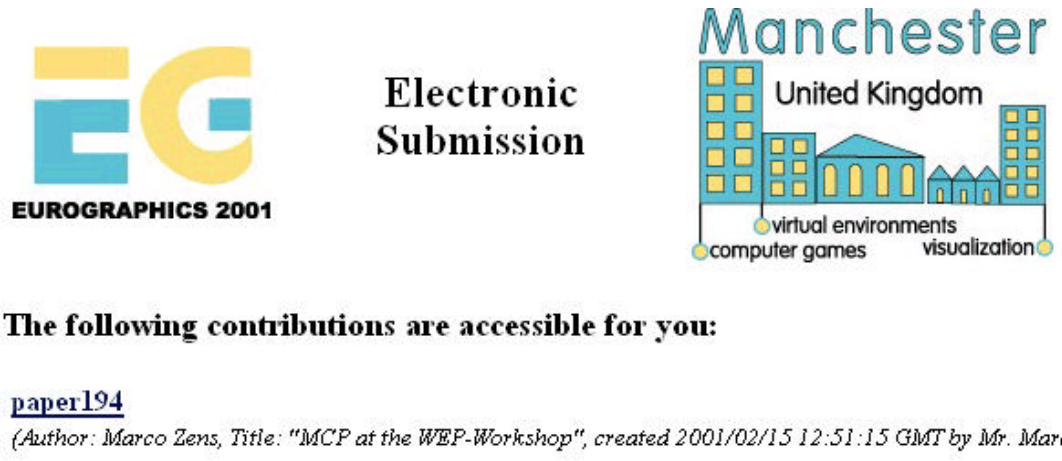


Figure 5.13: Here a list of accessible contributions is shown, as displayed to authors (i.e. their previous submissions), reviewers (i.e. the assigned papers) or the Administrator.

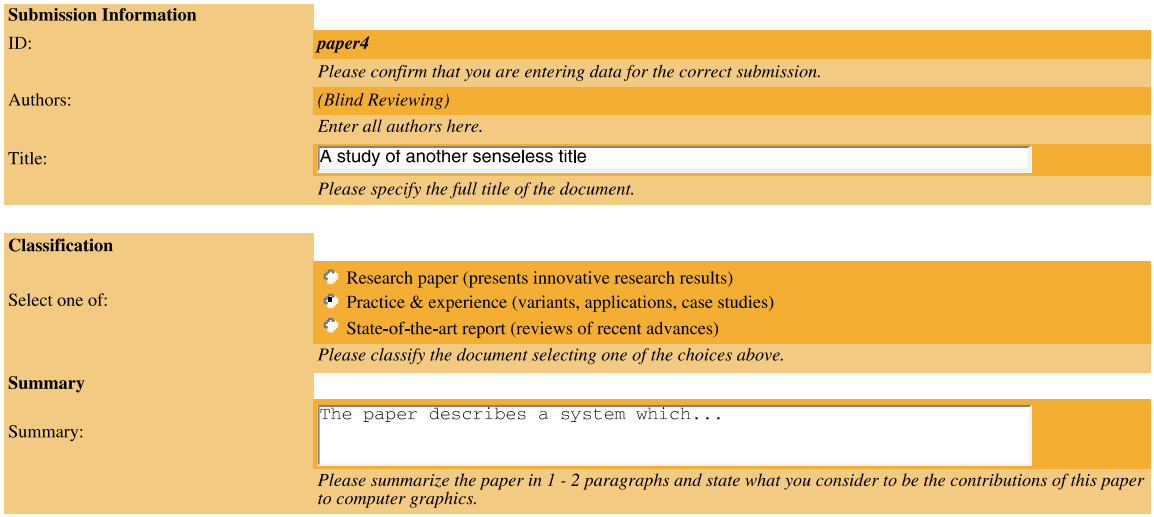


Figure 5.14: This figure displays an excerpt of the Web form which a reviewer can use to enter his comments.

the authors followed the instruction not to include it in the actual document).

All reviewers of a contribution are listed at the bottom of the submission's Web page (see Figure 5.16). In the example, all three reviewers have already provided their reviews. The overall result is presented and the reviews can be displayed in a printable, well-structured form (see Figure 5.17). This can also be extracted to the file-system for all papers in one step by a script, which is very useful for the Programme Committee Meeting or for internal discussions. Additionally, the reviews can be exported to XML (identical to the offline review form) as well as to CSV (e.g. for importing it into a spreadsheet software) format and finally they can be edited online, using the form also applied for entering them (see Figure 5.14).

This display of the reviewers and their reviews is an example for the context sensitive (or to be precise, depending on the access rights of the current user) menus of the *MCP* system. The one presented in the figure would only be visible to the Administrator or the listed Primary Reviewer in *Review Phase II*. An ordinary reviewer (i.e. a Secondary Reviewer or the Primary Reviewer in *Phase I*) will only get to see his own name. The author will see no reviews or reviewers at all.

Also during the reviewing or in preparation of the Programme Committee Meeting, the PCC can check the current status using several of the available online statistics. Figure 5.18 shows an example from the *Eurographics 2002* conference. It lists how many reviews are already available for how many and for which submissions, additionally giving their overall results. The example is from a later period of the *review phase* because at least two reviews are finished for all the papers, except for three papers (which actually were invalid submissions and thus not reviewed at all).

Finally, the reviewing is finished, the authors submit final versions of the accepted contributions and the proceedings are produced with the help of the tools and the metadata included in the *MCP* system. The last figure presented here, Figure 5.19, shows proceedings on three different media (from three different events) which are usually published: the online version of the *EG2000* conference in the *Eurographics Digital Library*[13], a page from the printed proceedings of *Eurographics 2001* [CR01] and as latest item the CD-ROM created for *EG2002*.

[13] http://diglib.eg.org/EG/DL/Conf/EG2000

```
<?xml version="1.0" encoding="iso-8859-1"?>
<review>
<!-- REVIEWER'S REPORT FORM -->

PAPER NUMBER        <num> ... </num>
AUTHORS             <au> ... </au>
TITLE OF THE PAPER  <ti> ... </ti>

<!-- ============================================================
 Please classify the paper selecting one of the choices below
   R - Research paper (presents innovative research results);
   P - Practice and experience (variants, applications, case studies);
   S - State-of-the-art report (reviews of recent advances);
-->

CLASSIFICATION   <cls> ... </cls>

[ ... abbreviated ... ]

<!--
Please rate the submission using one of the following integer values:
  0 - totally unacceptable   --   9 - excellent
-->

OVERALL RECOMMENDATION   <ova> ... </ova>

ADDITIONAL COMMENTS FOR THE AUTHORS (will be passed along anonymously)
<aucomment>
 ...
</aucomment>

</review>
```

Figure 5.15: Optionally, a reviewer can create his review offline by filling out an XML document. This figure shows an abbreviated version.

Figure 5.16: The reviewers assigned to a submission are listed in the Web browser. If reviews are already available, they can be displayed or modified.

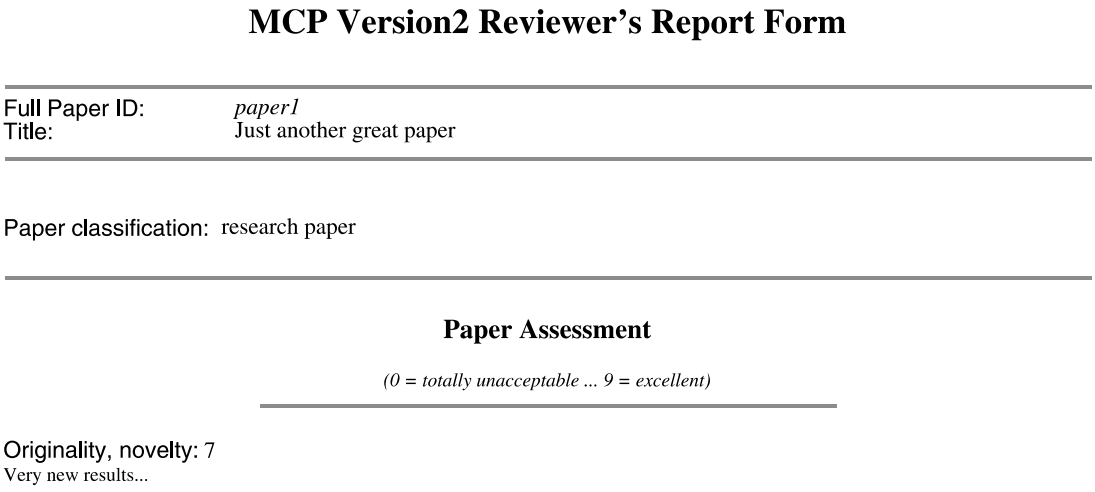


Figure 5.17: This is a small part of the reviewer's report form as displayed in the Web browser.

Eurographics 2002
Electronic Submission

Review Status

paper

Total amount:*232*

no reviews available for:*3*

- paper7
- paper26
- paper293

---

one review available for:*0*
<submission ID> <overall results>

---

two reviews available for:*3*
<submission ID> <overall results>

- paper39 5 5
- paper257 8 8
- paper259 7 4

Figure 5.18: This figure shows some more online statistics (abbreviated). The Programme Committee Chair can easily check how many and which reviews are still missing (including a quick overview of the current results).

5.6 Summary

The *Managing Conference Proceedings* system as discussed in this chapter is based heavily on the implementation details and on the architectural frameworks developed during the *GoldenGate* (see Chapter 3) and the *EG Online* (Chapter 4) projects, namely on the combination of standard components by Web technology and on Web Services. *MCP* can thus be understood as a follow-up (or even a conclusion) and it serves as a third application scenario about digital documents. In this case, a complete electronic document creation and publication workflow is implemented and/or supported.

The underlying problem which led to the *MCP* project is introduced in Section 5.1: producing proceedings of scientific events is continuously getting harder (e.g. larger events, shorter deadlines, multimedia documents, cross-media publishing...). Additionally, in the traditional (i.e. mostly paper-based) workflow the whole work is centered around the Programme Committee Chair, being responsible for communication and document transfer, for submission as well as review and for the creation of the proceedings. The only solution is the use of *Conference Support Software* (see below) supporting the complete production pipeline, which will as an ultimate goal improve the scientific quality of the event. The *Managing Conference Proceedings* system is such a software.

In contrast to 1998, when the *MCP* project was started, today many events make use of software tools and several companies are offering commercial support, but many of these only implement one or several isolated steps of the complete necessary workflow process. In Section 5.2 some well-known systems are described, namely *START* (used at several hundred events already), *WitanWeb* (e.g. used by the *WWW* conferences), *CyberChair* (concentrating on a reviewing process based on patterns identified) and the *Microsoft Conference Management Toolkit* (as an example of a system from a *large* company).

The research programme *Global Info* (see in Section 5.3) has been run by the German BMBF in the years 1996 – 2000. The goal was to develop scientific information systems to support all aspects of a

Figure 5.19: Here some final results of publication workflows supported by *MCP* are presented: proceedings have been published in the *Eurographics Digital Library*, in printed form and on a CD-ROM.

Digital Library, therefore all participants in the information distribution process should work together. All projects with a similar focus were grouped into sub-programmes like the one called *WEP* (dealing with the active part of document creation), of which *MCP* was a member.

The goal of *MCP* was to implement a complete digital publication workflow for a scientific conference, including support for submission and review, templates and guidelines for the authors as well as the final publication. The system should be freely available, it had to handle multimedia documents and to be fully usable through the WWW (see Section 5.4). We developed *MCP* in close contact with the users and made a real-life test at a big event with every subsequent prototype.

Our general idea is presented in Section 5.4.1. The new workflow had to be based on the four aspects *digital multimedia documents*, *automation* of routine work, efficient *tools* for all participants and *Web Services* for access and administration in order to support the input, the managing and the output as well as other, secondary tasks. This was achieved by using a *Hyperwave Information Manager* as Multimedia Document Management and simple Workflow Management system to become the central component and the main workhorse of the new, complete electronic workflow. Other techniques used include the programmable layout-templates of *HIS*, (CGI-) scripts in *Python*, emailing, document templates and a pre-press toolkit (see below).

A detailed description of all workflow elements containing four different phases was given in Section 5.4.2. First, an instance of the *MCP* system is configured by by the so-called *administrator* (i.e. possibly one of the event organizers) in the *installation phase*. Then, during the *submission phase*, authors can register to the system. They receive an account which enables them to make a new submission (entering metadata and uploading documents) or to access previous ones. When the *review phase* is started by the *administrator*, author access is disabled and the access for reviewers is enabled. For this to work, only a list containing the reviewers and the assigned submissions as well as an email template has to be provided. The reviewers can read the metadata and download all documents. Then they are asked to make their reviews, either through an online WWW form or by sending a filled out text document (i.e. XML format). After the programme committee meeting the final *publication phase* begins, during which the authors receive the review results (containing anonymous comments). Final versions are collected from the authors and assembled to the different publications with the help of included tools. All the time, the *administrator* can access the complete metadata, all material and dynamically generated statistics to simplify his job.

An important part in the final step of the publication chain is played by the pre-press tools included in the *Managing Conference Proceedings* system, see in Section 5.4.3. Technical progress allows it today to create professional looking publications on your own (instead of having to go to a publisher). This can offer advantages in cost, in time and in flexibility. However, several problems have to be solved, like the assembly of many documents, a consistent page-numbering, the normalization of page sizes and content offsets and the adding of copyright information or a logo. When PDF documents are used, this can be achieved easily and efficiently using four plug-ins (i.e. `CGFpagesizer`, `CGFnormalize`, `CGFpaginate`, `CGFbranding`) and one stand-alone tool (`CGFmerge`) which we have developed for *Adobe Acrobat*. They have been used successfully to complete all pre-press steps for the printed proceedings of the *Eurographics 2002* conference.

Prototypes and more advanced versions of the *MCP* system have been used successfully at several conferences and workshops during the last years, some of them being real stress-tests due to the number of submissions and their size. See the list in Section 5.4.4 and Table 5.1. Generally the system worked well but we made several observations leading to extensions and modifications. As discussed in the section, a HTTP-upload has proven to be superior to FTP-upload. Network problems have to be considered, as well as user errors (on one hand a freedom of choice and assistance as well as sometimes a gentle push on the other hand is necessary). At every run, users will miss some functions, which requires the system to be flexible and extensible (what *MCP* has proven to be). Finally, PDF or PostScript are the most suited formats. They can be created by everyone, but some pitfalls remain (like the embedding of

fonts and images).

For the *Eurographics 2002* conference a "production release" of *MCP* was ready and complete (see in Section 5.4.5). Of course, more improvements and extensions are already thought of. For example the review assignment could be prepared automatically (by generating proposals), indexes for the publications (or even the conference programme) could be generated from the available metadata and the flexibility of the system could be improved (like allowing different review forms). However, the future (even extended) application of *MCP* is already guaranteed, because the *Eurographics Association* has decided to use it for their conferences and for five workshops a year. They have just started to take the publication and the distribution of scientific documents in their own hands and they have acknowledged the fact that an Event Support Software is necessary to achieve this on a high quality level.

The final section of this chapter (i.e. 5.5), except for this summary, contains screenshots and documents from the previous applications of the *MCP* system to illustrate a complete example run. In form of a "slideshow" from the points of view of all participants the complete workflow (see also above) is shown.

Chapter 6

Conclusion

During this dissertation, the modern use of digital documents was discussed. We learned how they can be used to improve existing traditional workflow processes. As a start, definitions for important terms as well as document formats and procedures currently in use were presented. Later, three projects were described which serve as application scenarios on how to create, how to access, how to manage and on how to publish digital documents successfully. In this last chapter, it will be summarized what was achieved (see in Section 6.1 below) and what still has to be done in the future (Section 6.2).

6.1 Thesis Summary

Although digital documents play a steadily increasing role in our modern society, there is still some discussion going on about how to define terms like "Digital Document" (or even "Document") and "Digital Library". In the Definitions 1 – 2 in Chapter 2 my personal point of view of a (generalized) "Document" as a unit of information was presented, which then makes a "Digital Document" only a special case. Finally, a "Digital Library" is defined as a virtual collection of generalized digital documents, *including* the necessary methods to manage them and to work with them (see Definition 3).

An important question is how digital documents should be stored (e.g. when building a digital library). Generally, the "best" format does not exist. Instead, the one best suited should be chosen, depending on the document content, the environment and whether it is a structured or a presentation-oriented unit of information. Keeping this fact in mind, several widely used document formats were presented in Chapter 2. The most interesting ones currently being the markup languages (like XML). Besides other advantages, they are able to store metadata together with the primary content. Anyway, before a document can be stored, it has to be created first. This can be done either by retro-digitization or by creating a new one, possibly in one step also enriching the document with electronic added value. For the actual storage, Document Management Systems are very useful, allowing efficient access, searching and unambiguous identification, especially over the Internet and the World Wide Web.

The first of three application scenarios presented in this thesis concentrates on the creation of and the access to structured digital documents. During the research project *GoldenGate* (see in Chapter 3) a prototype system of a complete electronic workflow for the submission, the managing and the approval of grant proposals at the *Deutsche Forschungsgemeinschaft* (*DFG*) was designed, developed and evaluated. Here, a traditional paper-based process was replaced by a digital document workflow through the application of an Information Management System (i.e. *Hyperwave Information Server*) and the *Microsoft Office* tools.

The system implemented is based on a new paradigm, namely the combination of standard office tools with and also through Internet technology. By the combination of components which are already available and also *used* in the destination environment, the users gain for example a smooth integration, a guaranteed functionality and an automatic improvement for free whenever one of the components

evolves. As far as software development and maintenance as well as the necessity to train the users are concerned, these goals are achieved with less effort compared to building a complete new system from scratch or deploying an existing one. The validity of this result has been shown by our prototype during evaluations at our cooperation partner and during a real-life test run (see in Chapter 3).

Another profitable approach regarding the use of digital documents is the application of Web Services. In Chapter 4 the *EG Online* project was discussed, which is the set of Web Services of the *Eurographics Association*. The Association's goal is to offer distributed collaboration, information and administration to members, officers and others using only a Web browser and the Internet. This is achieved by combining several databases and an Information Management System with WWW front-end (again, *Hyperwave Information Server*, see also in Chapter 3). What once started as a simple Web server delivering information, has successfully evolved to the central tool of the Association's daily work over the last years. This includes many new services like, for example, a digital library. Today, *EG Online* is still hosted, administrated and extended by our group and it has not reached its limits yet.

The third and final application scenario about the modern use of digital documents – discussed in Chapter 5 – consists of the research project *Managing Conference Proceedings* which was a part of the *Global Info* programme. Here, the focus lies on a complete solution for the creation and the managing of digital documents as well as a publication workflow. *MCP* is a conclusion to the previous projects, because it builds on the developments and on the experiences already made. As a result, Web Services, standard tools, an Information Management System and Internet technology are the building blocks of the new system and the applied digital publication process.

In more detail, *MCP* is a software system and also an approved workflow to handle and to support the submission, the reviewing and the publication of papers at scientific events. This system differs from other Conference Support Software in that it targets the complete publication chain, from templates and guidelines for the authors to the final cross-media publishing. The new system is based on the four aspects *digital* multimedia documents, *automation* of routine work, *tools* for all important tasks and *Web Services* for access and administration. A multimedia database is used as central component and main workhorse, in contrast to burdening the event organizers themselves with the mentioned jobs like in traditional environments. All participants in the workflow directly "communicate" with an Information Management System, thus relieving the Programme Committee Chair from most of his work. Finally, by the use of the included tools, the accepted documents can be easily assembled and directly sent to the printer or put onto a CD-ROM etc. (i.e. cross-media publishing).

MCP has been successfully used at several large conferences and workshops already in its prototype stage, where it fulfilled our expectations and those of the users (i.e. authors, reviewers and organizers). After some iterations and extensions the stability of a production release is achieved. The system and the inherent workflow will be used by the *Eurographics Association* for future events and their publications on a semi-commercial basis.

6.2 Future Work

Though digital documents, the techniques and the procedures to use them as well as the application of digital workflows have improved dramatically over the last few years, there is still a lot of work to do. For example, it is still an important problem to enable and to achieve a consistent digital cooperation. Though many internal document workflows – be it in companies or in government institutions – are now completely "digitized", often a media break occurs as soon as a document leaves the Intranet: it is printed, forwarded on paper and then manually entered into a computer again. Therefore, what is missing in many places is the use of well-defined, standardized and structured formats for information interchange, including metadata. The new markup languages and Web Services are a great step forward as far as this problem is concerned, because they offer at least a technical framework for its solution.

Another severe problem lies on the side of the authors of digital documents (i.e. during document creation). Although both the technical knowledge and the necessary formats to create well-structured documents have been available for some time, the software (like word-processors) to do this efficiently and easily (meaning without having to make a MSc in Computer Science first) is still missing. The most widely spread tools to create text documents still work presentation-oriented. However, structured documents are undoubtedly necessary for an error-free and deterministic automated processing of digital information.

The research projects discussed during the course of this thesis tried to offer solutions to these and other problems on a small scale, i.e. for a specific problem domain and/or a small group of users. To this extent, they were very successful. Both the *GoldenGate* (see in Chapter 3) and the *Managing Conference Proceedings* (Chapter 5) research projects are finished. A prototype of the *GoldenGate* system was completed, but it will never be used nor developed further because the *Deutsche Forschungsgemeinschaft* decided against its application and by now time has passed on. Nevertheless, the basic ideas and the implementation framework (like the combination of office components with Internet technology) have been reused in later projects. For example, the *MCP* system is based on these, and here the situation is completely different (as far as the system's usage is concerned).

The *MCP* software has just reached a production release. Nevertheless it is already used heavily for more than two years. Through a contract with *Eurographics* its future on a semi-commercial basis is guaranteed for some time. Of course, there is also much room for extensions and improvements. Some of these were already described in Section 5.4.5. Another option is to include the second big achievement of *GoldenGate*: the extraction of structured data from presentation-oriented documents (i.e. the process named *XML Up-Translation*). It should be possible to automatically get much of the metadata which authors currently have to enter manually during submission (like author information or the title of their paper).

As far as the *EG Online* project is concerned (see in Chapter 4) the future is also clearly visible. The *EG Online* system is currently the tool of choice for the daily administrative work of the *Eurographics Association* and it will remain this way for quite some time. Be it in one form or another, Web Services are the only way to organize a distributed collaboration of volunteers on a non-profit and a free of charge basis. The clock cannot and will not be turned back. To the contrary, I am quite sure that more and more functionality will be added and more tasks will be processed digitally in the years to come.

List of Figures

List of Tables

Bibliography

[Ado98] Adobe Systems Inc. *PostScript Language Reference*. Addison-Wesley, 3rd edition, 1998. 10, 12, 107

[Ado01] Adobe Systems Inc., San Jose, CA. *Acrobat Interapplication Communication Overview*, October 2001. Technical Note #5164. 104

[AGH00] Ken Arnold, James Gosling, and David Holmes. *The Java^{TM} Programming Language*. Addison-Wesley, 3rd edition, June 2000. 106

[Ald92] Aldus Developers Desk. Tag Image File Format (TIFF) specification, June 1992. Revision 6.0. 11

[App00] Apple Computer Inc. *QuickTime File Format*. Vervante Inc., October 2000. 14

[BC93] Tim Bienz and Richard Cohn. *Portable Document Format Reference Manual, Adobe Systems Incorporated*. Addison-Wesley, June 1993. 8, 10, 107

[BHST95] R. Bentley, T. Horstmann, K. Sikkel, and J. Trevor. Supporting collaborative information sharing with the WWW: The BSCW shared workspace system. In O'Reilly and Associates and Web Consortium (W3C), editors, *World Wide Web Journal: The Fourth International WWW Conference Proceedings*, pages 63–74, Cambridge, MA, 1995. O'Reilly & Associates, Inc. 42

[BLF99] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper, San Francisco, September 1999. 16

[Bou97] Thomas Boutell. PNG (Portable Network Graphics) specification. RFC2083, March 1997. Version 1.0. 12

[Com87] CompuServe. *Graphics Interchange Format (GIF) – A standard defining a mechanism for the storage and transmission of raster-based graphics information*. CompuServe Inc., June 1987. 8, 11

[CR01] A. Chalmers and T.-M. Rhyne, editors. *Eurographics 2001 proceedings*, volume 20-3 of *Computer Graphics forum*, Manchester, UK, September 2001. Eurographics Association, Blackwell Publishers. 105, 114

[DCM02] Dublin Core Metadata Initiative (DCMI). `http://dublincore.org`, 2002. 19

[DD93] C. J. Date and Hugh Darwen. *A Guide to The SQL Standard*. Addison-Wesley, third edition, 1993. 28

[DHW97] M. Dörr, H. Haddouti, and S. Wiesener. The german national biography 1601 – 1700: Digital images in a cooperative cataloging project. In *Proc. Advances in Digital Libraries (ADL'97)*, Los Alamitos, CA, 1997. IEEE Press. 22

[Die92] Die Deutsche Bibliothek, editor. *Maschinelles Austauschformat für Bibliotheken: MAB*. Dt. Bibliothek, Zentrale Bibliogr. Dienstleistungen, Frankfurt a.M., 1992. ISBN 3-922051-41-3. 19

[DS02] G. Drettakis and H.-P. Seidel, editors. *Eurographics 2002 proceedings*, volume 21-3 of *Computer Graphics forum*, Saarbrücken, Germany, September 2002. Eurographics Association, Blackwell Publishers. 105, 108

[FE00] Dieter W. Fellner and Albert Endres. *Digitale Bibliotheken*. dpunkt.verlag, Heidelberg, 2000. 4, 5, 6, 18, 109

[Fel97] D. W. Fellner. DFG Schwerpunktprogramm Verteilte Verarbeitung und Vermittlung digitaler Dokumente. *Informatik Forschung und Entwicklung*, 12(1):38–42, 1997. 58, 60

[FGKM00] C. Fuß, F. Gatzemeier, M. Kirchhof, and O. Meyer. Inferring structure information from typography. In *Proceedings of the 8. International Conference on Digital Documents and Electronic Publishing (DDEP2000)*, München, 2000. Springer-Verlag. 24

[Fox93] Edward A. Fox (ed.). Source book on digital libraries. Technical Report 93-35, Dept. of Computer Science, Virginia Tech, Blacksburg, VA, 1993. 4

[Fur98] Betty Furrie. *Understanding MARC Bibliographic: Machine-Readable Cataloging*. Cataloging Distribution Service, Library of Congress, 5th edition, 1998. 19

[FZ99a] Dieter W. Fellner and Marco Zens. Electronic submission, managing and approval of grant proposals at the german research foundation based on standard internet and office tools. Technical Report TUBSCG-1999-02, Institut für ComputerGraphik, TU Braunschweig, 1999. 35

[FZ99b] Dieter W. Fellner and Marco Zens. Management and workflow of electronic documents using a 2nd-generation WWW-server. In Paul de Bra and John Legget, editors, *Proceedings of WebNet 99 – World Conference on the WWW and Internet*, volume 1, pages 354–359. AACE, October 1999. ISBN 1-880094-36-3. 40, 41, 96

[FZ00] Dieter W. Fellner and Marco Zens. Electronic submission, managing and approval of grant proposals at the german research foundation based on standard internet and office tools. *J.UCS Journal of Universal Computer Science*, 6(3):356–366, Mar 2000. ISSN 0948-6968. 35

[FZ01] Dieter W. Fellner and Marco Zens. Managing Conference Proceedings. Technical Report TUBSCG-2001-02, Institut für ComputerGraphik, TU Braunschweig, 2001. 70, 94

[Ger01] Rich Gerber. SoftConf.com: software for conferences – START. `http://www.softconf.com/START`, September 2001. 90, 91

[GH00] M. Gross and F.R.A. Hopgood, editors. *Eurographics 2000 proceedings*, volume 19-3 of *Computer Graphics forum*, Interlaken, Switzerland, August 2000. Eurographics Association, Blackwell Publishers. 105

[Glo00] Global Info Consortium (GIC). Global Info – the german digital library project. `http://www.global-info.org/wasistgi/index.html`, July 2000. 92

[GMS+98] H. Gladney, F. Mintzer, F. Schiattarella, J. Bescos, and M. Treu. Digital access to antiquities. *Comm. ACM*, 41(4):49–57, 1998. 22

[Gol90] Charles F. Goldfarb. *The SGML Handbook*. Clarendon Press, Oxford, 1990. 16

[Gre98] Gregory Grefenstette, editor. *Cross-Language Information Retrieval*, volume 2 of *The Kluwer International Series on Information Retrieval*. Kluwer Academic Publishers, Boston, March 1998. 29

[HBT97] Arthur S. Hitomi, Gregory Alan Bolcer, and Richard N. Taylor. Endeavors: A process system infrastructure. In *Proceedings of the 1997 International Conference on Software Engineering*, pages 598–599. ACM Press, 1997. 42

[HR00] Mark Hammond and Andy Robinson. *Python Programming on Win 32*. O'Reilly & Associates, 1st edition, January 2000. 50

[Hun99] Jane Hunter. Mpeg-7: Behind the scenes. *D-Lib Magazine*, 5(9), September 1999. 30

[Hyp01] Hyperwave AG, Munich, Germany. *Hyperwave Information Server: PLACE Workshop (Version 5.1)*, 2001. Hyperwave Guides. 39, 50, 86

[Ins00] Institute for Information Technology, National Research Council of Canada (NRC IIT). WitanWeb – Web-based Refereeing. `http://witanweb.iit.nrc.ca`, November 2000. 91

[Ise00] David Iseminger. *Com+ Developer's Reference Library*. Windows Programming Reference Series. Microsoft Press, September 2000. 44, 48

[ISO89] ISO. *ISO8879: Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*. ISO JTC1/SC34, October 1989. 16

[ISO90] ISO. *JPEG Technical Specification, Revision 5*. ISO JTCI/SC2/WG8 JPEG-8-R5, 1990. 8, 12

[ISO91] ISO. *ISO646: 7-bit coded character set for information interchange, Revision 3*. ISO JTCI/SC2 R3, 1991. 8

[ISO96] ISO. *ISO10179: Information technology – Processing languages – Document Style Semantics and Specification Language (DSSSL)*. ISO JTC1/SC34, 1996. 16

[ISO98] ISO. *ISO8859: 8-bit single-byte coded graphic character sets*. ISO JTCI/SC2/WG3, 1998. 8

[ISO99a] ISO. *ISO11172: Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video, Revision 2*. ISO JTC1/SC29 R-2, March 1999. 14

[ISO99b] ISO. *ISO11172: Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*. ISO JTC1/SC29, March 1999. 13

[ISO99c] ISO. *ISO8632: Information technology – Computer graphics – Metafile for the storage and transfer of picture description information – Part 1: Functional specification*. ISO JTC1/SC24, December 1999. Edition 2. 12

[JFS95] Charles E. Jacobs, Adam Finkelstein, and David H. Salesin. Fast multiresolution image querying. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM Press, 1995. 30

[KC01] F. Kurth and M. Clausen. Full-text indexing of very large audio data bases. In *110th Convention of the Audio Engineering Society*, Amsterdam, May 2001. Convention Paper 5347. 30

[KM97] Peter Kent and Kent Multer. *Official Netscape JavaScript 1.2 Programmer's Reference*. Ventana Press, 1997. 39, 50, 86, 96

[KMS93] F. Kappe, H. Maurer, and N. Sherbakov. Hyper-G: A universal hypermedia system. *Journal of Educational Multimedia and Hypermedia*, 2(1), 1993. 38, 96

[Knu86] Donald E. Knuth. *TEX: The Program*, volume B of *Computers & Typesetting*. Addison-Wesley, Reading, MA, 1986. 9

[Koh95] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995. 30

[Lam85] Leslie Lamport. *LATEX – A Document Preparation System*. Addison-Wesley, Reading, MA, 1985. 10, 107

[LCG+00] Steve Lawrence, Frans Coetzee, Eric Glover, Gary Flake, David Pennock, Bob Krovetz, Finn Nielsen, Andries Kruger, and Lee Giles. Persistence of information on the web: analyzing citations contained in research articles. In *Proceedings of the ninth international conference on Information and knowledge management*. ACM Press, 2000. 31

[LGB99] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999. 27

[LV01] Carl Lagoze and Herbert Van de Sompel. The Open Archives Initiative: building a low-barrier interoperability framework. In *Proceedings of the first ACM/IEEE-CS joint conference on Digital libraries*. ACM Press, 2001. 28

[Mau96] Hermann Maurer, editor. *Hyper-G/now Hyperwave – The Next Generation Web Solution*. Addison-Wesley, Harlow, England, 1996. 38, 96

[Mic99] Microsoft Corporation. Rich Text Format (RTF) specification, version 1.6. `http://msdn.microsoft.com`, May 1999. 9, 62

[Mic02] Microsoft Research. Microsoft Conference Management Toolkit (MS CMT). `http://cmt.research.microsoft.com/cmt`, September 2002. 92

[Mil98] Eric Miller. An introduction to the resource description framework. *D-Lib Magazine*, May 1998. 19

[MM-02] MM-WEP Teilnehmer. Abschlussbericht zur Sonderfördermaßnahme WEP – Projekt MM-WEP Multimediale Werkzeuge für elektronisches Publizieren. `http://vhb.fh-regensburg.de/wep/berichte.htm`, June 2002. 93

[NF98] E. J. Neuhold and R. Ferber. Scientific digital libraries in Germany: Global-Info, a Federal Government initiative. *Lecture Notes in Computer Science*, 1513, 1998. 92

[Nie00] Oscar Nierstrasz. Identify the champion. In N. Harrison, B. Foote, and H. Rohnert, editors, *Pattern Languages of Program Design*, volume 4, pages 539–556. Addison Wesley, 2000. 92

[Obe99] Carsten Oberscheid. XML up-translation. Master's thesis, Computer Graphics, TU Braunschweig, December 1999. 24, 54, 62

[Odl99] A. M. Odlyzko. Competition and cooperation: Libraries and publishers in the transition to electronic scholarly journals. *Journal of Electronic Publishing*, 4(4), June 1999. 22

[Pas99] Norman Paskin. DOI: Current status and outlook. *D-Lib Magazine*, 5(5), May 1999. 31

[RF01] Diann Rusch-Feja. The Open Archives Initiative – new access to scientific publications? *Bibliothek: Forschung und Praxis*, 25(3):291–300, 2001. 28, 109

[RJ99] Guido Van Rossum and Fred L. Drake Jr. *Python Reference Manual*. iUniverse.com, February 1999. Release 1.5.2. 48, 68, 85, 96

[RJB97] Jim Rumbaugh, Ivar Jacobsen, and Grady Booch. *Unified Modeling Language Reference Manual*. Addison Wesley, 1997. 37

[SABS94] Gerard Salton, James Allan, Christopher Buckley, and Amit Singhal. Automatic analysis, theme generation, and summarization of machine-readable text. *Science*, 264:1421–1426, 1994. 23

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983. 23, 26

[SM94] K. Sollins and L. Masinter. Functional requirements for uniform resource names. RFC1737, December 1994. 31

[Sno99] Rick Snodgrass. Summary of conference management software. `http://www.acm.org/sigs/sgb/summary.html`, January 1999. 90

[Sta00] Richard van de Stadt. Cyberchair: A web-based groupware application to facilitate the paper reviewing process. `http://www.cyberchair.org/wbgafprp.pdf`, May 2000. 91

[The91] The Unicode Consortium. *The Unicode Standard – Worldwide Character Encoding*, volume 1. Addison-Wesley, Reading, Mass., 1991. Version 1.0. 8

[The98] The VRML Consortium Inc. *ISO14772: Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language – Part 1: Functional specification and UTF-8 encoding*. ISO JTC1/SC24, June 1998. 13

[The00] The LaTeX3 Project. LaTeX: A document preparation system. `http://www.latex-project.org`, August 2000. 10

[The02] The European Association for Computer Graphics. The Constitution of the Eurographics Association. `http://www.eg.org/EG/Docs/Misc/Constitution.html`, August 2002. 66

[Tho00] Stephen A. Thomas. *SSL & TLS Essentials: Securing the Web*. John Wiley & Sons, February 2000. 86

[W3C98] W3C CSS and Formatting Working Group. Cascading Style Sheets, level 2 (CSS2) Specification. `http://www.w3.org/TR/REC-CSS2`, May 1998. 16

[W3C99a] W3C HTML Working Group. Hypertext Markup Language (HTML) 4.01 Specification. `http://www.w3.org/TR/html401`, December 1999. 16

[W3C99b] W3C Semantic Web Activity. Resource Description Framework (RDF). `http://www.w3.org/RDF`, 1999. 18, 19

[W3C00a] W3C HTML Working Group. XHTML 1.0: The Extensible HyperText Markup Language. http://www.w3.org/TR/xhtml1, January 2000. 18, 109

[W3C00b] W3C XML Working Group. Extensible Markup language (XML) 1.0 (Second Edition). http://www.w3.org/TR/2000/REC-xml-20001006, October 2000. 17

[W3C01a] W3C SVG Working Group. Scalable Vector Graphics (SVG) 1.0 specification. http://www.w3.org/TR/SVG, September 2001. 12, 18

[W3C01b] W3C XML Schema Working Group. XML Schema Part 0: Primer. http://www.w3.org/TR/xmlschema-0, May 2001. 18

[W3C01c] W3C XSL Working Group. Extensible Stylesheet Language (XSL) Version 1.0. http://www.w3.org/TR/xsl, October 2001. 17, 109

[WCO00] Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl*. O'Reilly & Associates, 3rd edition, July 2000. 67

[Wel84] Terry A. Welch. A technique for high-performance data compression. *IEEE Computer*, 17(6), June 1984. 11

[WF00] Ian H. Witten and Eibe Frank. *Data Mining*. Morgan Kaufmann, Los Altos, US, 2000. 26

[WHK97] M. Wahl, T. Howes, and S. Kille. Lightweight Directory Access Protocol (LDAP). RFC2251, December 1997. Version 3. 39

[WKLW98] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin Core Metadata for Resource Discovery. RFC2413, September 1998. 19

[WM99] Norman Walsh and Leonard Muellner. *DocBook: The Definitive Guide*. O'Reilly, October 1999. 25

[Zak00] Diane L. Zak. *Visual Basic for Applications*. Course Technology, 1st edition, June 2000. 47, 51